Bachelor project

# 27.04

I’ve decided to use nodeJS and expressJS for the backend.  
I considered using firebase but unfortunately there’s no way to store data internally and that would be a problem as we’re handling sensitive data from the patients.

They’ll be used to build a RESTfull API to communicate with a MySQL database (see if maybe it would be better to save data directly in HUG temp database).

# 28.04

We’ve talked with Mr. Gluck about the scope of the project and redefined it a little by adding IBeacon detection and location within the HUG mainframe.  
To do that I must use Bluetooth LE technology which is not available by the means of a web app.

We decided to switch to native applications both on android and ios.  
I now have to search and find the best suited framework between ionic phonegap and apache cordova to build my application.

We talked with Mr.Ehrler about my gantt chart and what where the main issues that had to be fixed. Basically I had to add some simple technical description a little more in depth than what I did for each main task.

# 29.04

I’ve decided to use ionic for several reasons.

Phonegap is basically apache cordova with addons for adobe softwares which I have no interest using.

Apache cordova is great and would have worked just fine but I would have had to redo a lot of work and spend time learning their API.

Why Ionic was a better fit and cordova ?  
Ionic is based on angularJS, which is great because it was part of what I learned to use during my semester project so I would be able to save a lot of time using it.

Both have live preview of the in development app through native applications found on the app store and play store.

Both are based on web technologies and have the ability to work with IBeacon.

# 30.04

I’ve finished updating my gantt chart and sent it to Mr.Gluck and Mr.Ehrler.

I’ve also started working on ionic and was able to display the list of interventions ordered by time and type as I did for the semester project.

I’ve decided to use a modular structure for my project because of several advantages it gives for big projects development and code maintaining.

Here’s a list of those advantages:

* 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.

# 01.05

Official state vacation (fête du travail).

# 04.05

I experienced some issues with github refusing to work properly as I changed my username. Fixed it by reconfiguring my default credentials on all my devices.

Didn’t work so much this day…

# 05.05

I’ve almost finished the Interventions view, which lacks a few functionalities that I can’t implement for now. I need to ask a few questions to Mr. Ehrler about the nurses schedule.

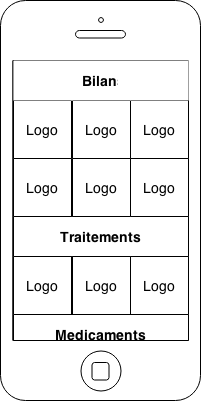
I also had some issues with the data.xml file (the one containing all the sample dev data from the HUG), it was not imported properly on the ionic view app which is suppose to be the live preview of my app for Mr Gluck and Mr. Erhler.  
I decided to host this file on my own private server and allow CORS (**C**ross-**O**rigin **R**essource **S**haring) on this specific repository.

It works fine now.

I’ve added a header to the view which allow the user to delete acts or reach the menu by sliding his finger on the left direction or clicking on the hamburger logo.

What’s left for the intervention view ?

* Automatic scrolling to the current time appropriate acts
* Only show the acts to be performed on the nurse’s shift (can’t be done now)

I’ve also thought of a 2nd layout presentation for the intervention view.

I could use tile design :

# 06.05

I’ve asked 3 other bachelor students what they think about the interface I built for the interventions view and then to compare it with the old one.

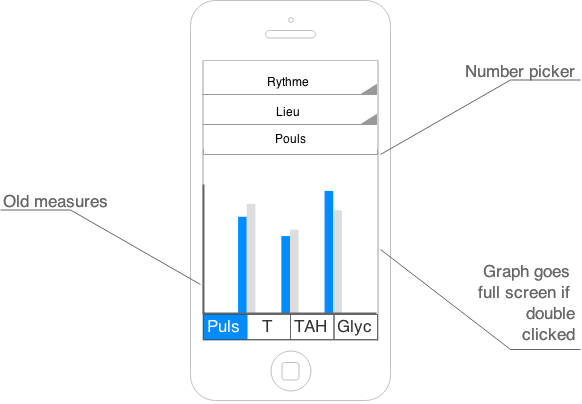
I’ve received good feedback from it, they understood easily how it would work and were familiar with the data representation which they said was clearer than the old one.

I’ve also showed them my wireframe for the alternative view of the intervention page, they said I should keep the text otherwise it would be too hard to remember which logo represent which intervention

I’ve wireframed the vital sign view.

# 07.05

I’ve had some trouble adding a second module/page to the project as I forgot to include the .js file in the main index.html (dummy error but took me about 1 hour t Suo figure out !).  
I’ve wireframed the vital sign view.

My aim for the day is to correct the few remaining problems in the intervention view and to implement the static part of this view.

I’ve discussed with Frederic about my interface for the intervention view. He told me it should look more like the demo application with icons. My main goal should be to stick as much as possible to it.

I’ve decided to reproduce the intervention view so I’ll be taking some time to make the changes.

Frederic explained me that the nurses shift is handled on the server side and therefore I don’t need to implement anything about it.

# 08.05

I’ve made all the changes necessary for the intervention view to match the intervention view of the sample android app including icons.

Here’s a screenshot of what it looks like right now.

I’m a few days late compared to my planning because I didn’t asked Mr. Ehrler to check the wireframe of the view I wanted

# 11.05

I’ve had significant troubles putting on the navigation between to views.

Today I’ve studied the way routing works on ionic, here’s most of what I learned.

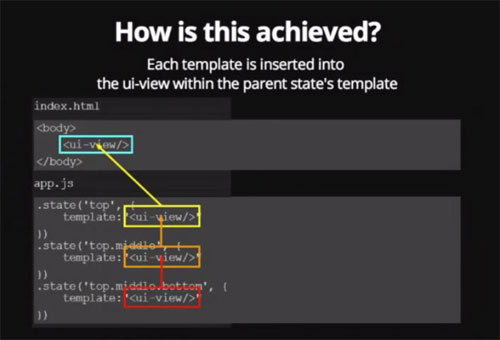
Ionic works with states instead of routes, they present some differences.

|  |  |
| --- | --- |
| **Angular with ngRoute** | **ionic with ui.router (state machine)** |
| Flat hierarchy  If you want the details of a group of act you need a new URI for instance :   * /interventions {view intervention} * /interventions/details {view details intervention}   Each URI theoretically reloads the whole view (maybe some fixed content wont change but we will reload most of the dynamic part). | Nesting and inheritance If you want the details of a group of act you can create a sub-state for instance:   * /interventions – {view intervention, view interventions.details}   We can just load what’s changing on the page. |
| The name of a route is it’s url | The name of a state is an actual name |
| You can navigate to a route only by it’s url | You can navigate to a state by it’s name with the command $state.go(stateName) you can also go to your parent state by using $state.go(‘^’) |
| Single view using ng-view (lots of reloading) | Several views nested into each other, which means more focused reloading. |
| A state can populate any view within its hierarchy. | Populates the view you get. |

Some more information about nesting:

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

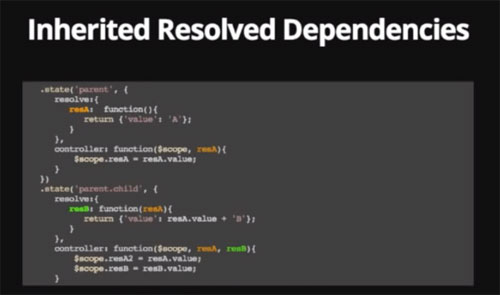
Each template (part of a page that can be on another file) is loaded into a <ui-view> or a <ion-nav-view> tag within the parent state template.



How does inheritance work within nested state?

A nested state inherits the scope and the methods of it’s 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 :



Here we can see that 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"

}

}) ;

Simply use the property data to set those custom data specific to a state and it’s children.

It then can be accessed using this variable $state.current.data.customData1

You can change state with a custom directive ui-sref which stands for   
state reference => smart anchors. It’s to be used instead of the usual href.

This allows to change the url without changing them in the application.  
It will generate the 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 allow us to tie a state to a specific url.  
if state ‘sample’ has url ‘/sample’ and I create a state sample.child and define it’s url to /child then the url of the child will really be /sample/child.  
This can be avoided by using the ‘^’ before the url => ‘^/child.

# 18.05

Haven’t work this Monday.

# 19.05

I’ve linked all interventions icons to the UI using a large ngSwitch.

I’ve replaced the ngSwitch directive by a simple scope variable that I forged with a switch before loading it on the page.

I’ve added the possibility to take notes, it will then delete the element from the list and save it into an array which will be later on submitted to the server.

The same functionalities are to be found on the child state/view details.

Problem remains with the deleteElem function when in details view deleting an element doesn’t work anymore.

The main view is almost over, once this problem has been solved and the view validated by Mr.Erhler on Thursday, I will then move on to the vitals measurement view.

# 20.05

I’ve solved the problem causing the details view’act not to disappear after notes are taken.

I’ve also solved the problem causing the return link not to work. Apparently, a child state cannot load it’s parent, it can only be loaded into it.

As a consequence I’ve created an abstract state interventions in which I load the patient interventions or a group of interventions.

An abstract state is a state that isn’t loadable, it’s many uses like defining a common controller, or resolve common variables.

# 21.05

Today I will discuss with Frederic about the finished interventions view, see if there’s anything to add, then I will validate the wireframe I’ve done for the vital signs view.

Once that’s done I will start implementing the view for the vital signs and try to finish it by the end of the day.

Once that will be done I will have accomplished the 2 most complicated view of the application, it took me a lot of time for several reasons, mostly because of the time I invested which wasn’t close to 8h a day.

Another reason is that I had to master a new framework, Ionic (version 1.0 just went live which is great => more stability and native scrolling for ios and android).

The rest of the development should be way faster for the simple reason that those 2 views helped me understand the functioning of ionic and how I should think and build my app.

During this first month I’ve done a lot of restructuring and rewriting because the more I was learning the best I understood how I should proceed.

That is the reason why I’m not worried about how late I am compared to my gantt chart. I know the rest will be faster even if not without problems.

I’ve discussed with Frederic about my view and asked some questions :

* Would it be useful to save the seconds at which an operation has been validated

Why not

* What can help me determine if an act can be validated or not ?

If it’s an act group or it it’s too early in the day then the user can’t validate it.

* What does force valid means exactly ?

Force the validation of an act even if too early.

* Surveillances => What does the number of surveillances means ?

The minimal measure of vital signs that have to be performed, inside the xml I can find the exact measures needed.

Other important points we discussed:

Acts in timeframes like Matinéé, diurne, nocturne should always come first while we’re in the time frame, for instance, matinée will come first while we’re between 6 am and 12 am then will stay at 12 am.

The Reserve section should always be visible and follow the title.

I should handle this with a small JSON config file to make it easy to modify.

Possibility to access the different actions for an act by a slide left on the act, which would unveil them. Often seen on IOS and android.

In the vital signs view, the possibility to zoom and unzoom on the graph should change the scale to allow a wider view of the results.