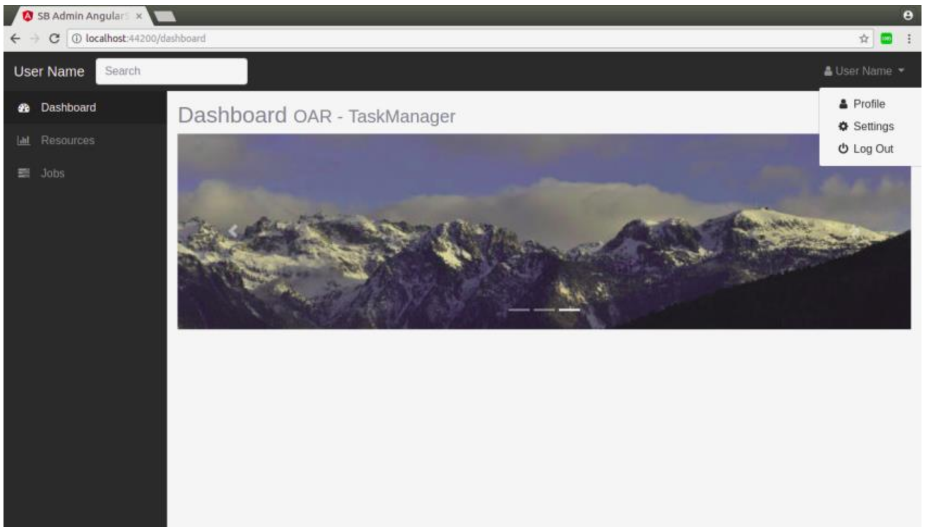
RICM4 2017-2018

# Dashboard for Task Manager

*Development of a dashboard for the Resource and Task manager, OAR*



BELGUENDOUZ Sekina - LARNICOL Titouan

## TABLE OF CONTENT

# INTRODUCTION

## Presentation of the project

The project consisted in updating a single page application (SPA), OAR-Skylight originally developed with Angular 2. Its aim was to create an easier way to use the Task Manager, OAR. Before its development, it was only accessible via command lines, which are not very intuitive or practical. This is the reason why OAR-Skylight was created. However, few years later, Angular proposed a new version of its framework: Angular 5, allowing better performances and optimization of the applications.

## Presentation of the technologies

### *OAR / OAR-Docker:*

"*OAR is a versatile resources and tasks manager (also called a batch scheduler) for HPC clusters and other computing infrastructures*". This definition is from oar.imag.fr, its official website.

OAR aims to manage the tasks submitted by the users, according to their needs and the resources available. It is based on a database and retrieves information when needed. OAR takes care of the scheduling of the different tasks, depending on the parameters entered by the user. All of its features are available through command lines, which execute independent module.

To use OAR's API, we needed to be registered users or admins, which we aren't. Therefore we had to use OAR-Docker. It allowed us to create our own OAR cluster with Docker images on our personal computers. We had a small cluster with a server, a frontend and virtual nodes to test our SPA.

### *Angular:*

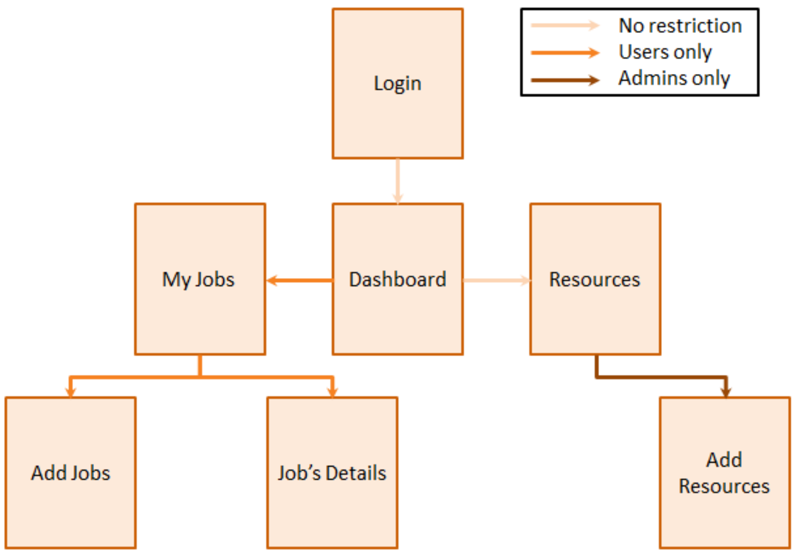
Angular is a framework based on TypeScript. It is used to develop cross-platform applications for web, web mobile… It eases the developer's work to create SPAs, offering not only great performances but also simple templates to build frontend web apps. It comes with http services, form handlers and much more. Through its evolution, Angular framework became faster and more compact, offering more and more services along the way. The most recent version, Angular5, propose a lot of new features: date and currency pipes, compiler improvement… But the most interesting one is the HttpClient module; it allows the developer to make API calls more easily.

### *OAR-Skylight:*

OAR-Skylight is a project developed by Remi BOUGUERMOUH in 2016 in Angular2. It was the perfect technology to create it with at the time. Unfortunately, Angular2 was quickly followed by two new versions: Angular4 and 5. And even though, this project is only two years old, it rapidly became obsolete.

## HOW IT WORKS

### Architecture

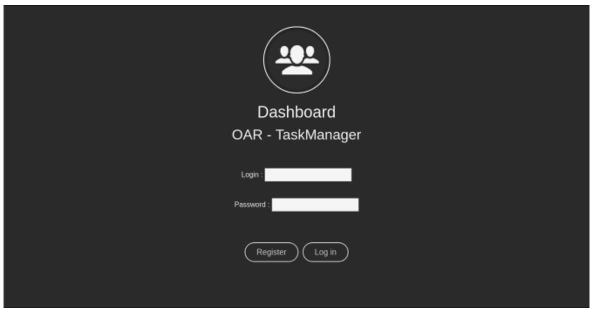


We decided to base our SPA on OAR-Skylight, as well as SB-Admin-BS4-Angular-5, a template offered by Angular5. Therefore, it simply consists, like Skylight, in 7 different pages:

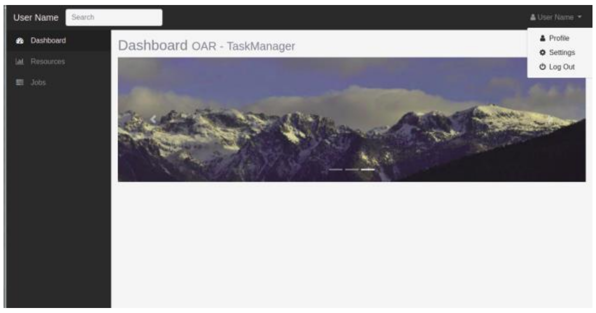
- Login: Allows the user to log in or to go on the 'home' page, *Dashboard*.
- Dashboard: Allows the user access the other pages, if he has the proper authorisation.
- Resources: Presents a list of the current resources in the cluster.
- Add Resource: Allows the admins to add a new resource.
- My Jobs: Shows to the user his submitted jobs.
- Job's Details: Shows to the user all the details of one of his submitted jobs. (Not in the final product)
- Add Job: Allows the user to submit new a new job.

## Template

First of all, we decided to create a simple template, which would then allow us to make basic call to the API.

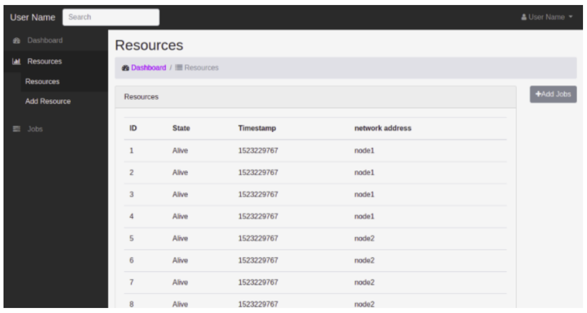


So we took the two pages *Login* and *Dashboard* and their components from SB-Admin-BS4-Angular-5. And then we modified them a little bit to suit them to our needs. For *Login* we only changed the name of the application.

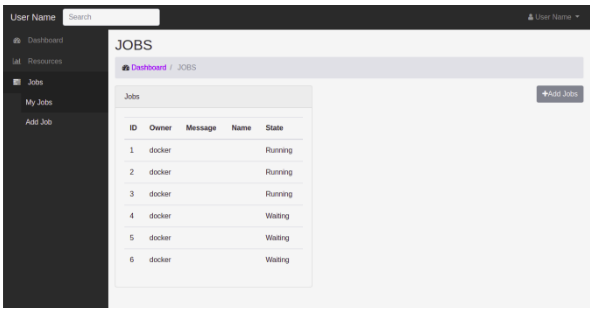


However, for *Dashboard* we modified a lot more. We removed the unused pages in the sidebar and added submenus to the *Resources* and *My Jobs* sections. We then removed the unused options in the header leaving only the *Profile* tab. And finally, we delete all the unwanted components of the *Dashboard* page, leaving only a sliding banner.

After those minor changes, we created two pages *Resources* and *My Jobs.* Those two pages created the same way. They are composed of a grid and a button. The grid is filled with, respectively, all the current available resources, or the submitted jobs.



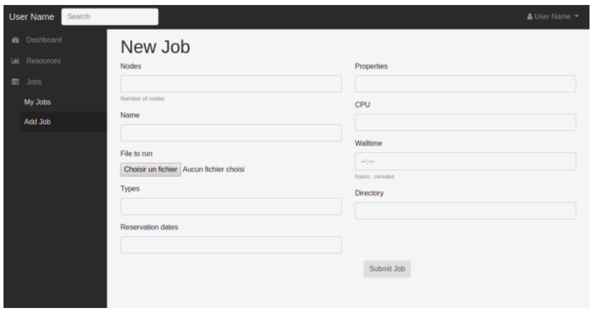
| ID | State | Timestamp | network address |
|---|---|---|---|
| 1 | Alive | 1523229767 | node1 |
| 2 | Alive | 1523229767 | node1 |
| 3 | Alive | 1523229767 | node1 |
| 4 | Alive | 1523229767 | node1 |
| 5 | Alive | 1523229767 | node2 |
| 6 | Alive | 1523229767 | node2 |
| 7 | Alive | 1523229767 | node2 |
| 8 | Alive | 1523229767 | node2 |

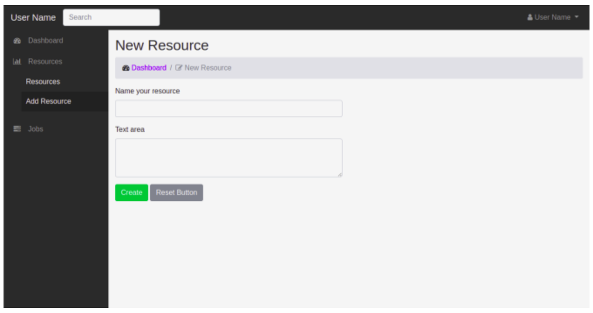


| ID | Owner | Message | Name | State |
|---|---|---|---|---|
| 1 | docker | | | Running |
| 2 | docker | | | Running |
| 3 | docker | | | Running |
| 4 | docker | | | Waiting |
| 5 | docker | | | Waiting |
| 6 | docker | | | Waiting |

Then we have a button leading to new pages, either *New Resource* or *New Job*. Thanks to those screenshots, we clearly see the submenus we talked about. They allow the user to jumps between *Resources* or *My Job* to *New Resource* or *New Job* more efficiently.

And finally, we created two forms for *New Resource* and *New Job*. Unfortunately, we only had time to do completely *New Job* with all the parameters possible. For *New Resource*, we only have the base, as can see in the screenshot.





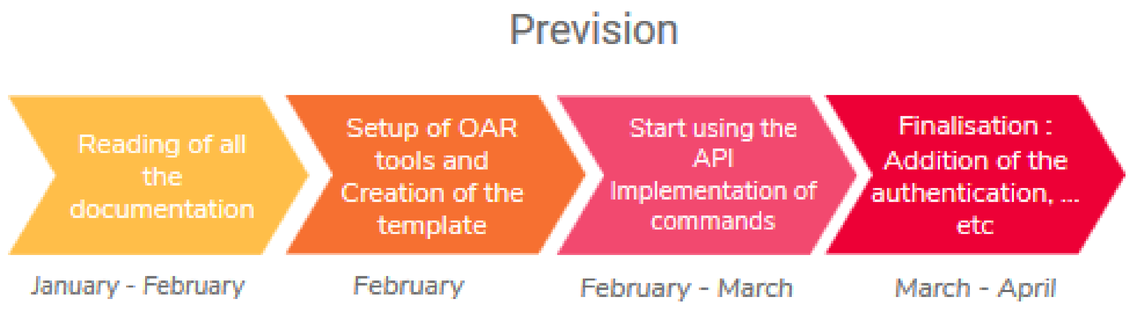
## API's calls

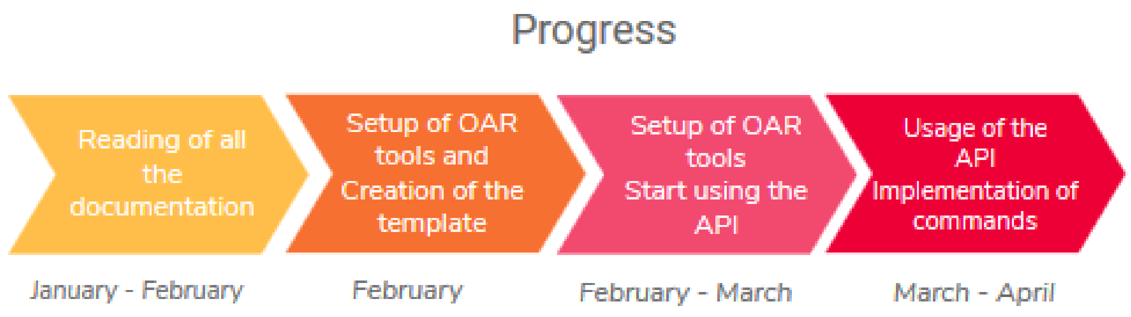
It took us a lot of trials and errors to access the API from OAR-Docker and we finally manage to use it only by the end of the project. Yet, we manage to make some calls in the program.

We are able to display a list of all the current resources and all their details (page *Resources*) and we can display all the submitted jobs and their details (page *My Jobs*).We tried to use the API to submit new job but we had an issue with the authentication. It completely stopped our progression (we couldn't create job or distinguish the users) leading to our unfinished project.

# CONCLUSION

## Progression


Prevision
Reading of all the documentation
January - February
Setup of OAR tools and Creation of the template
February
Start using the API Implementation of commands
February - March
Finalisation : Addition of the authentication, ... etc
March - April


Progress
Reading of all the documentation
January - February
Setup of OAR tools and Creation of the template
February
Setup of OAR tools Start using the API
February - March
Usage of the API Implementation of commands
March - April

We couldn't keep up with our prevision mainly because the set-up of OAR-Docker and the access to the API made it impossible. We had to face a lot of issues we wouldn't have been able to resolve without our supervisor.

## Thoughts

This project was very interesting because we, as student, often forget the job of a developer is also to maintain and update codes. We had to understand and filter the already existing code to our needs, and not only restart for scratch.

The usage of all those different technologies was also beneficial, even though it made the developing environment very complex. It made us realise how difficult it can be to deploy software.

However, we also has this feeling disappointment and unaccomplished. We didn't really had time to really code because of the far too complex developing environment. It took us so much time to set everything up and yet it feels like we didn't use it at all.

## Future

To conclude, we will share our take on the remaining development. With the new type Observable of Angular5 and AsyncPipe, will probably replace the notion of store of the previous project. Indeed, the AsyncPipe used with an observable returns the latest value it has emitted. When a new value is change, the pipe marks the component to be checked for changes otherwise it will not reload the values. Moreover, the authentication is supposed to be far easier with Angular 5. The module LoginComponent is said to be all that is needed to create an authentication app.

## BIBLIOGRAPHY

https://oar.imag.fr/

https://angular.io/docs

https://github.com/oar-team/oar