Development of a SaaS as an IT trainee : final part

Introduction

It's already the end of my internship at Belighted, last weeks passed amazingly quickly.

I'd like to thank so much all the Belighters for their particularly warm welcome, and all the things and advice they gave to me.
Especially Philippe who was my internship mentor and taught me a huge amount of numerous things about good practises, design pattern, RESTful architecture, testing (a lot about testing), etc ...

It's kinda funny to take a look again at the code of my first school project in Rails. This is making me realize how much Belighted made me progress.

Let's now talk about what I did and learnt during these last weeks.

Implementing Pravatar on Scale

Scale is one of Belighted's project.
It's a resource managment system SaaS designed for digital agencies.

Scale has no avatar management except Gravatar, this is what led to the idea of Pravatar.
That idea is providing Scale with a separate avatar system, then re-use it on other projects needing avatars, and eventually making it available for external projects.

So once I made Pravatar working, and tested it on an other project including the dedicated **gem** and javascript **snippet**, it was the time to make the final test, running Pravatar on Scale !

The available tools

During this work on Pravatar, I built these tools in order to make clients able to use the services :

- **API** : An API following the JSON API specification.
- **Script** : Included in a `<script>` tag by the client, which can then use Pravatar feature simply by using some `data-purpose` attributes in the HTML.
- **Gem** : Working alongside the script, the gem is simplifying the process a step further, providing two simple, concise and configurable helpers for displaying and uploading Pravatar features.

Equipped with these tools, next step was to discover Scale codebase and figure out how to set up Pravatar.

Scale analysis

At first glance, Scale quite frightened me because it was something I never saw before.
I was expecting some simple Rails structure with just an app folder containing the `mvc` folders and nothing more, so I could just insert the snippet in the classic `application.html.erb` file, then put the gem in the gemfile and raise my hand telling I was finished with this.
How naive was I ! However, looking at it today, I'm **so much** more satisfied that it hasn't been that easy.

So, Scale was in fact an *AngularJS* single-page app served by the Rails backend app.
I didn't understand how exactly AngularJS is running with Rails in Scale, but I quickly understood that I would have to work only with Angular. So I took some days to learn Angular after analyzing the code.

The tools and Scale

Equipped with my new knowledge of AngularJS, I was able to review the usability of my tools inside the project.

Javascript Snippet

The main problem with my snippet was that I based all the functionnalities on the `$document.ready()` event, which was fine with a "stateless" web-app like my Rails demo, but totally **not** with an Angular "stateful" one !

So I updated that script by providing my functionnality inside a *Pravatar* object attached to the `window` by the script.
But it didn't worked well anyway, moreover it seemed quite ugly to me to mix up my script with angular and brutalize its logical process with my functions.

In conclusion, the snippet wasn't designed for that kind of project.
Even if I would have been able to make it work with Angular, I'm still not sure about how good it would have been.

The Rails Gem

My failure with the snippet was implying that I could not use the gem, because it was designed to work alongside it !

Anyway, even if the snippet would have been totally fine, the gem would have been constituing a source of issues.

The Pravatar gem has `JWT gem` as a dependency.
Trying to `bundle install` Pravatar gem on Scale led to a monumental amount of **sub-dependecies versions conflicts**, solving these conflicts by upgrading these dependencies could have made Scale totally unstable for whichever gem version update.

Pravatar API

This was the only option left to make Pravatar works on Scale, and probably the **best**.

If I managed to build a script using this API to run the service, the thing was totally achievable by using that API with AngularJS despite my lack of experience and my brand-new newbie knowledge about it.

Implementation

The only parts of Scale that I had to focus onto were the avatars.

As the avatars were only displayed (no update/upload options), I rapidly found out that all of the avatar displays on Scale were achieved by the same `directive`.

In order to use my Pravatar API, I had to build a `PravatarProvider` which would be initialized with the *apiKey*, the *projectName* and the *pravatarUrl*.
This provider would then expose two functions, a `load` one for loading the avatar from a profile id, and an `upload` one for uploading a new avatar with base64 image data and the profile's id.

As the display zone for avatars was perfectly designed, the result of a fetched Pravatar was marvelously fitting in it.
As the upload function wasn't implemented before, it took me a bit of time to design something intuitive for the users.
But with the advice of Simon, and re-using some *loading-effect* element of Scale, I managed to achieve a stylish upload feature on Scale.

Like usual, it seems really simple now, but it took me some time to figure out how Angular was working and become confident with it.

Conclusion

These fifteen weeks were really awesome, so much new experience and stuff to learn, some beers with the Belighters, being happy to go there every day !

I really hope that the Pravatar stuff will be useful to Belighted in some ways and that I achieved good work for them, as they gave me a lot in counterpart.

That experience confirmed my will to pursue in the web development area, and I feel really lucky to have made that internship in such a good company wich such nice people.

Thank you again Belighted !