Web Oriented Application

JeLisFusion

POLYTECH MONTPELLIER

Thomas Falcone
Informatique et Gestion
3ème année - 2017 / 2018

Table of content

Global presentation

JeLisFusion is a local bookshop project led by two students-in-law in the University of Montpellier. The idea is to increase access to culture for young generations. That's why, JeLisFusion is **entirely dedicated to children and teenagers**, offering classical child books and comics. Also, both booksellers decided to animate some **workshop to let people discover new field** (like Origami, the Writing of books and Reading Club).

In order to success, this project needs a **powerful tool for the client to have visibility on the bookshop**. That is why I decided to create a Web Oriented Application for this project. This represents a meaningful added value to the bookshop.

In this application you are able to **see available books** in the bookshop and **workshop animated in the days to come** (you can also **book a seat** for these workshops).

The booksellers have a **special access to the Manager Side** of the application. In this part, you can **add, modify or delete** books and workshops, as well as authors and publishers. This side of the project has been designed in order to **make management easier**. You can easily **see the remaining stock** you have for a book and the **remaining seats there are for a workshop**.

How to Access this project

The application of JeLisFusion is available at **https://jelisfusion.herokuapp.com**
You can create an account and see the User Side of the application or you can use the dedicated Manager account with this information :

Mail address admin@teapot.com
Password admin

Also, there's a git in order to download the code source of this project at https://github.com/ThomasF34/KidsWorld (you might see "KidsWorld" in several places : that's the former name of this project)

Architecture of the application

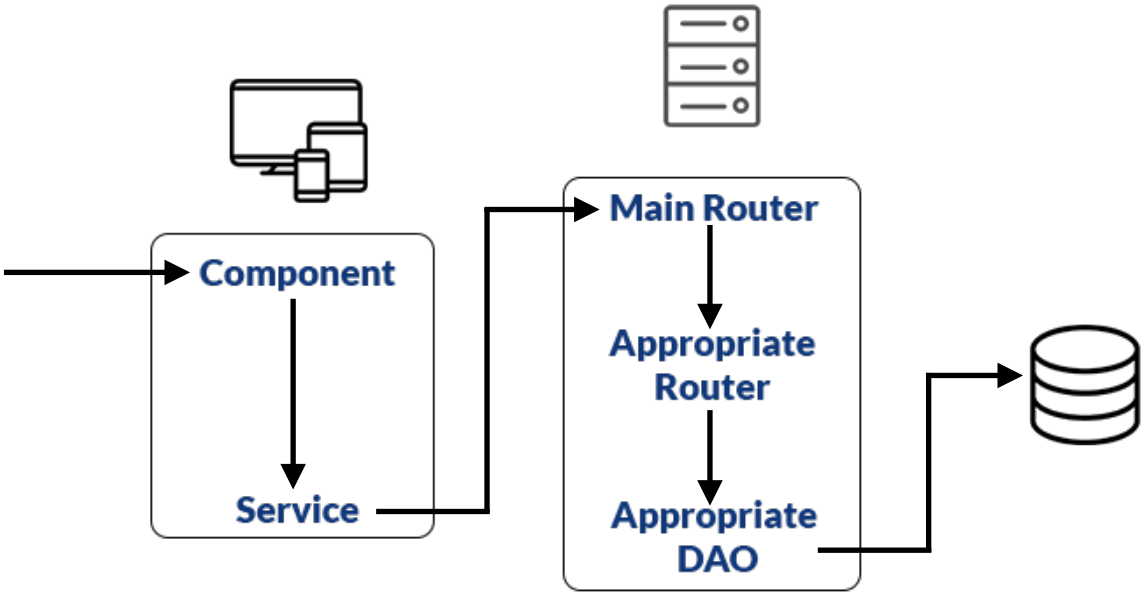
The application is base on a SOFEA architecture, **Service Oriented Front-End Application**. In concrete terms, this means that **most interactions are made on the client side** and the only interactions with the server are at the first connection (to gather all the components) and when you need to ask or send information to the database.

Client Side

On the client side, I used the **Angular 5** technology. Thus, you'll have access to the application throughout components. These are little part of a page that can have a unique behaviour and can be used in several pages. For example, a navigation bar is needed on each page. So the navigation bar is developed as a component that will be used for the entire application. It will have the same unique behaviour on each page since it is the same component.

Server Side

On the server side, I used **NodeJS** technology, which is able to route the request sent by the client side and to access the database. Plus, with this technology we can assure that the requests to the database are made by users with the appropriate entitlement. (A 'normal' user won't be able to create new books.)



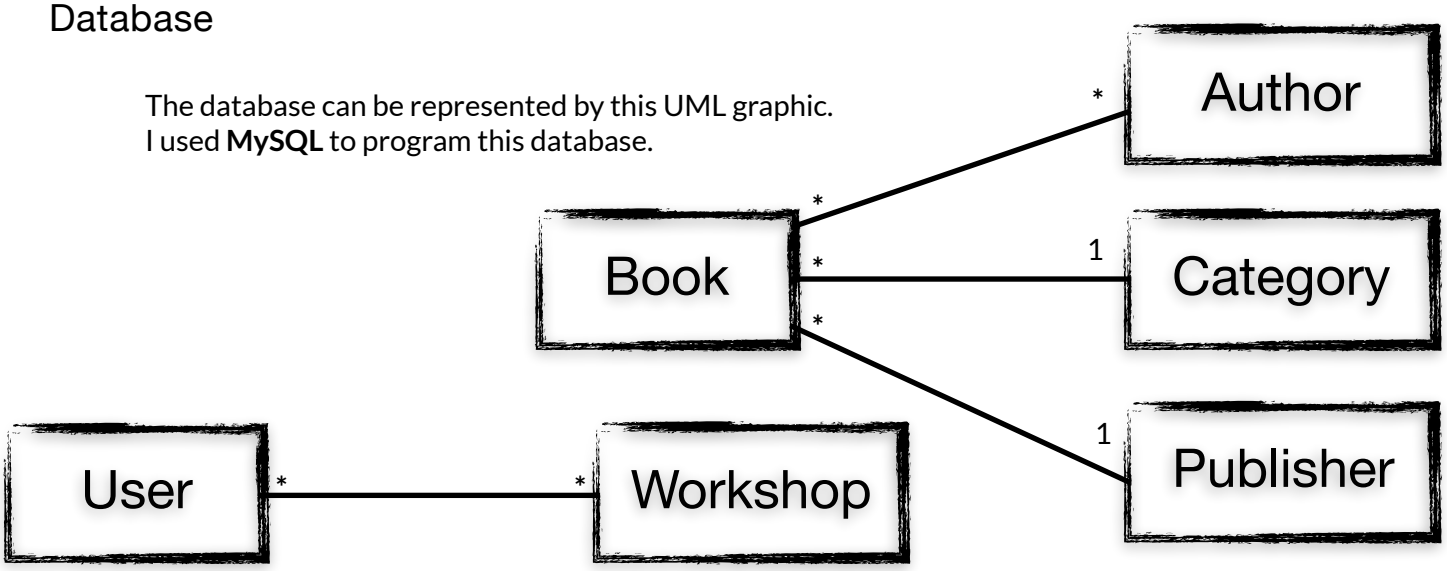
Authentication

I also used a module called **JsonWebToken** (JWT) that will generate, on the server side, a token with encrypted data when the user logs in or registers. Then this token is sent to the client side and stored in the local storage of the user (on its browser). Thus, when the user makes a request (to edit a book for example) we **intercept the request on the client side to add this token**. When the request arrives on the server, we can **easily check, by decrypting the token, if the logged in user is an administrator** or not. In this way, the server can return the appropriate HTTP code. Last step : according to the HTTP code received, the client side can redirect the user to another page (for example to the login page if the server says 401, which means it doesn't know who is connected).

Plus, for security purpose, I used **bcrypt**, to encrypt and decrypt passwords, in order not to store unencrypted passwords in the database.

Database

The database can be represented by this UML graphic.
I used **MySQL** to program this database.



Reasons to choose these technologies

The architecture made with the technologies NodeJS and Angular paired with MySQL brought some pros and cons that I balanced in order to verify that they are the appropriate languages for my application

Pros

The separation between Front-end and Back-end with Angular and NodeJS brings a significant advantage : the **fluidity of navigation between pages**. Since everything, except data requests, is placed on the client side, the navigation is really fast. Plus, with Angular's component, the page doesn't have to be entirely refreshed. When you navigate throughout the web application, only the component that changes are reloaded.

For a **scalability purpose, it is also a significant advantage**. Obviously when you have 10 000 users to satisfy at the same time, it's **easier to only handle data requests** instead of data and pages requests !

This architecture brings another benefit : Between front-end and back-end there is only one "tube of communication", throughout Services. All data requests are sent from front-end-coded services which make the trip of data-request much more understandable.

Cons

Using MySQL for the database can be a drawback for this project. During their navigation on the JeLisFusion application, user will **mainly read information**. So the main action will be getting information from the database, and in this field, **MySQL is not really scalable**. Choosing technology like NoSQL (Not only SQL), that are letting down the relational vision of classical database, would have been wiser, for a scalability purpose.

On the other hand, I choose MySQL anyway because it was **suitable to my data structure**. (Since NoSQL technologies has a strong-less definition of data schemas).

For client-side authentication I decided to **store a boolean in the local storage** of the user's browser. Thus, the Authentification Guard, which will verify this boolean, will let, or not, the user see a page. Of course, **aware of the danger of this solution**, I implemented **verification on the server-side** to be sure that the user is truly an administrator (and not someone who has changed the value of the boolean in local storage)

How to improve the application

To improve the application we could implement new functionalities like **making a reservation for a book**, maybe even paying online. We also could improve the actual functionalities of the app. For example, for now, there's no cover function. When you are adding a new book with the app you do not have the possibility to **join an image file** that represents the cover of the book. Alerting administrators about the stock of books would be a great improvement too.

On a technical level, we could improve the behaviour of angular's components. For now, they are not communicating. For example, when the server returns an error because you tried to add a book with a non-administrator account, or when your token is expired, you are redirected to the login page. When you arrive on the login page, **there is no message or alert informing you of what happened**.

Lastly, when administrators are adding books, it would have been great to **be able to create editors, authors and categories 'on the go'**. This would be a great work to do in order to discover **asynchronous request**.

Since I'll continue my work on this project, I'll implement these new features.

Retrospective of the project

What went wrong ?

Though NodeJS and Angular were two great technologies full of advantages, I was a total beginner with these languages. It has been a hard step to learn all the basics and use it on the project in a little amount of time.

In a short period, I realised how important it is to be organised ! If I had to start again this project from scratch, I'll make a detailed list of tasks to do, associated with time it could take to make them.

What went right ?

In this project I spent much time on a prior reflexion, unfortunately not on how to handle the short amount of time we had, but rather on how to structure the application. This helped me a lot, all along the project, because the organisation of the data was clear in my mind.

What I have learned ?

It has been a pleasure to see that every subject of the year was mixed in this project (from the analysis of a company's need to the elaboration of the structure of data). Plus, I'm now more self-confident about my ability to learn by myself. Since it's a very important quality nowadays, I'm glad I have been able to increase this criterion during this project.

Annexes

All modules that I used during this project are listed below. To discover and install them I used **npm**, a package manager for NodeJS.

Bcrypt - Encrypt and Decrypt password

Angular-calendar - Calendar used for the workshop listing
Date-fns - Calendar used for the workshop listing

Angular-datepicker - Used in form to choose workshop date and time

Express - Used to route requests in the server
Express-myconnection - Used to make request to MySql Database

JsonWebToken - Use to create and verify token to handle authentification