

Hackathon Day

The target of this test is to build a clone application of the famous platform 'Medium' using Angular (version 10) for the frontend, Symfony (version 4 or 5) based on API Platform and NodeJS for the backend connected to Relational database (SQL).

For this clone application, we have to type of users:

- Simple user
- Admin

A simple use, after signing-in, can create a new article, react to existing articles, leave a comment, edit his own articles, or even delete or unpublish one of them. Besides, the connected user can search for keywords inside the content of the articles or by tags. In addition, an admin has the right to execute any action on any articles **except deleting articles**.

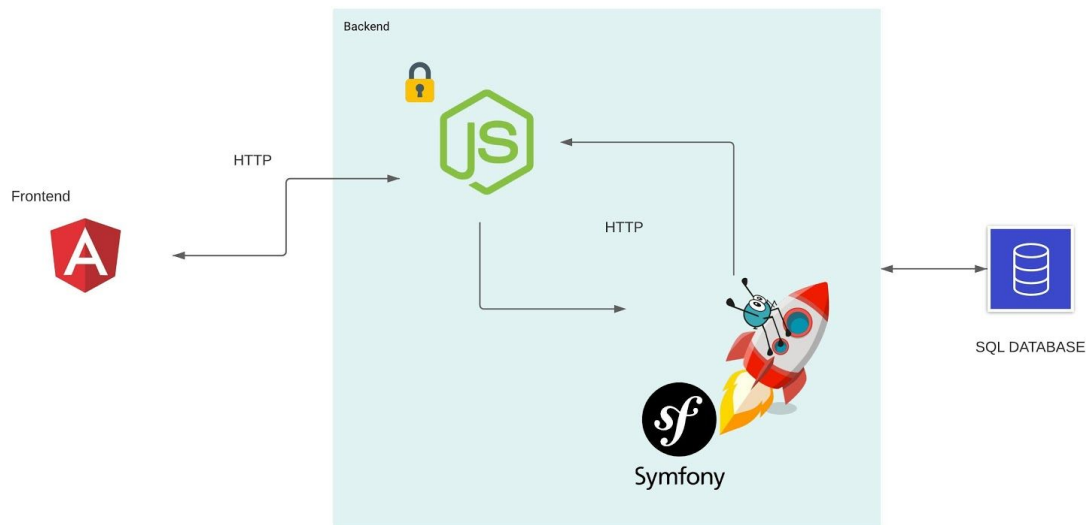


Figure 1: Technical stack

*All your applications have to be **dockerized**.*

A. Backend:

This part represents the core of the application, which is composed of two applications. The first one for authentication and authorization and the second one for providing the main endpoints of business logic.

1. Smart-Auth:

The aim of this microservice is quite simple, authenticate the users and authorize their actions. Smart-Auth will provide these endpoints that allow to:

- Create a new account (mention the role)
- Sign-in
- Change password
- Access to core endpoints (authorization)

As shown in Figure 1, every HTTP call from the front-end should be authorized by Smart-Auth. To build this microservice you have to use NodeJS. You are free to pick your framework (Express JS, Nest JS, Fastify, ...).

2. Smart - Doctrina:

This Symfony application is based on the API Platform. You have to implement the class diagram below in Figure 2.

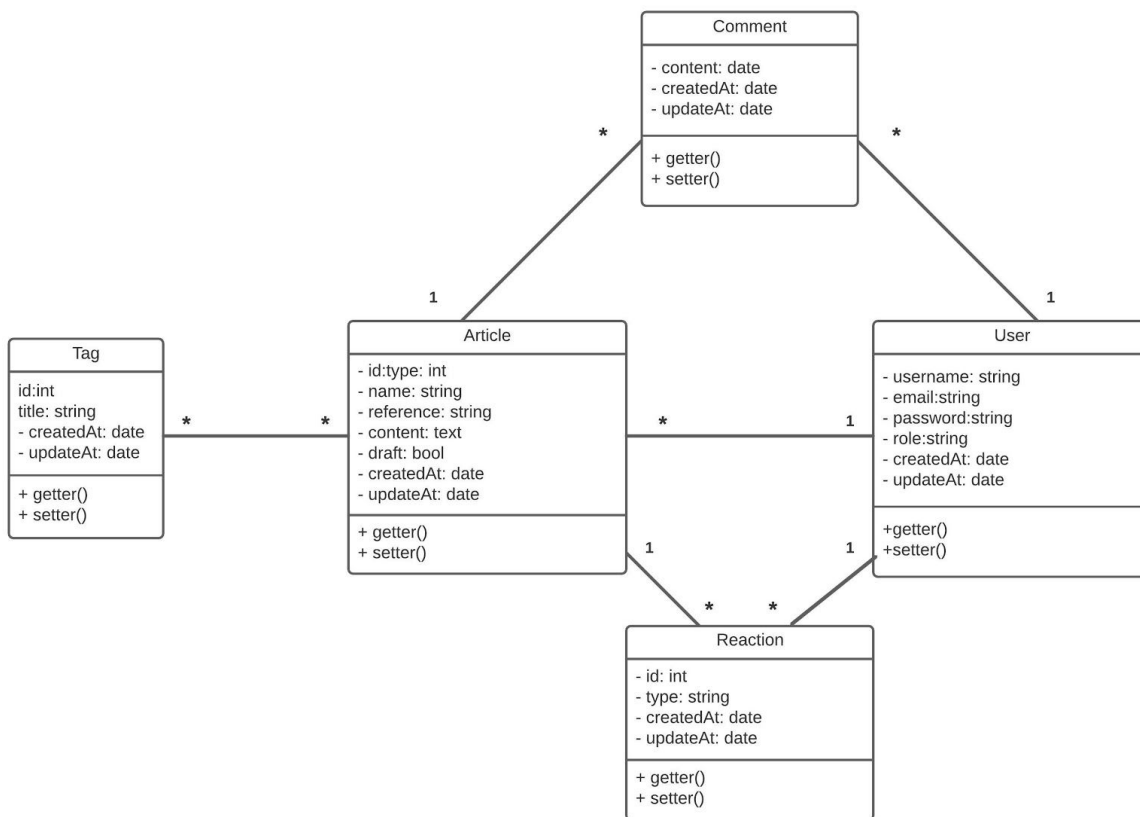


Figure 2: Class diagram

N.B: the user part should be managed with Smart-Auth.

You have to generate the CRUD endpoints for these classes: Article, Tag, Comment, and Reaction. And provide the endpoints for fetching articles by tags or by keywords.

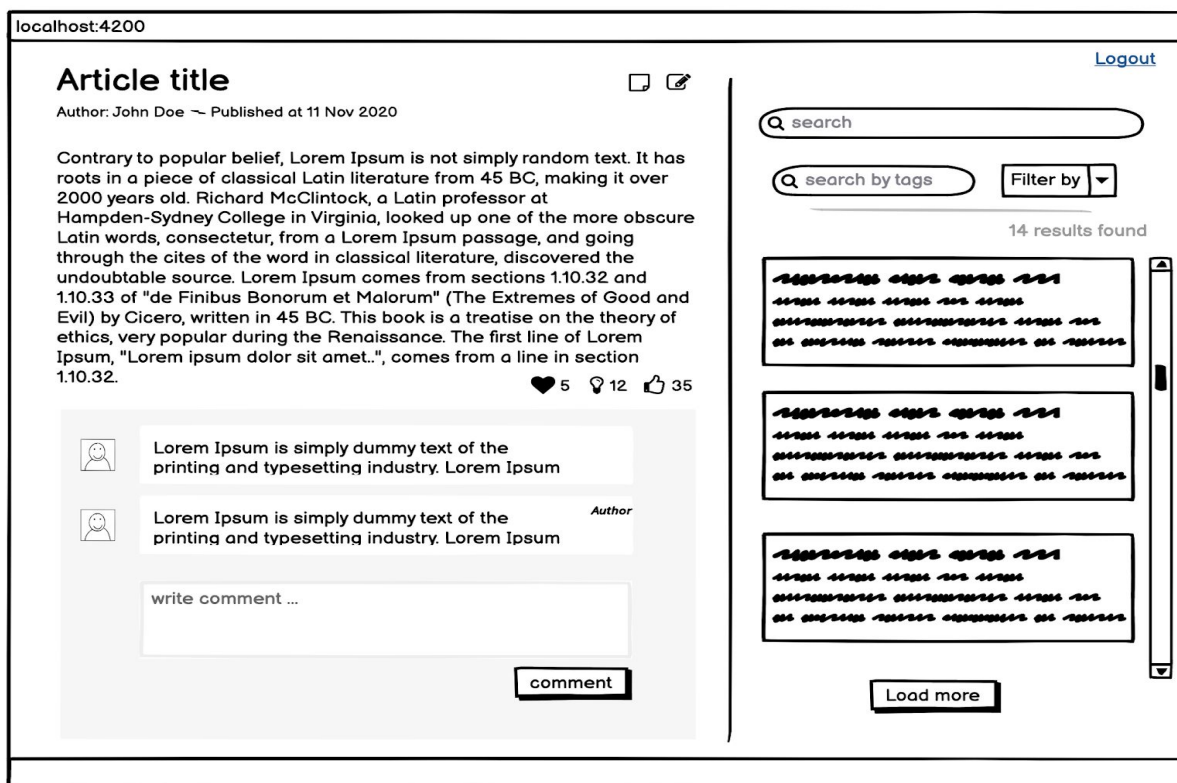
N.B: Articles and comments can be edited only by their creators or by an admin.

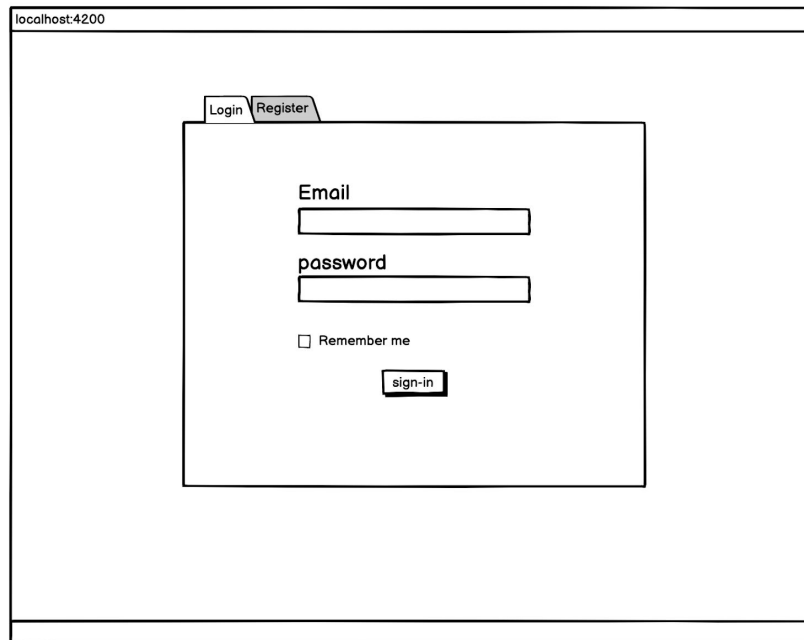
B. Frontend:

As mentioned above the frontend application will be coded with Angular with the implementation of the **SSR (Server-Sider-Rendering)** technique. Besides you have to build these interfaces which represent the use cases describes above.

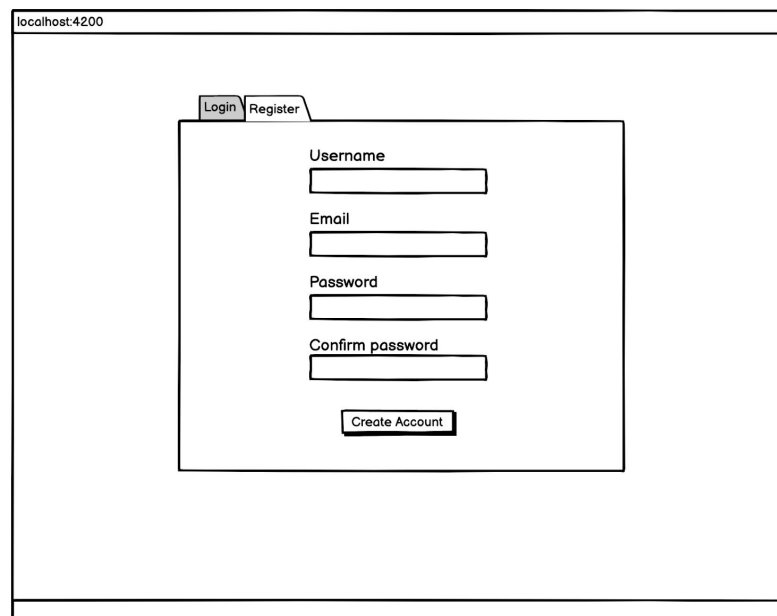
You can use any CSS framework or external libraries of your choice.

You have to build an interactive interface by consuming the necessary endpoints provides by the backend.





A web browser window showing a login form. The address bar displays 'localhost:4200'. The form is centered and contains a 'Login' tab and a 'Register' tab. Below the tabs are two input fields labeled 'Email' and 'password'. A checkbox labeled 'Remember me' is positioned below the password field. A 'sign-in' button is located at the bottom of the form.



A web browser window showing a registration form. The address bar displays 'localhost:4200'. The form is centered and contains a 'Login' tab and a 'Register' tab. Below the tabs are four input fields labeled 'Username', 'Email', 'Password', and 'Confirm password'. A 'Create Account' button is located at the bottom of the form.

Finally, don't forget to share your project on git repository with a readme file and a demo of your delivery (use Loom if you want, it's easy and simple), make semantic commits during coding (It helps the team to understand the way you work).

Bonus Points:

- Think to handle the cache for the backend's endpoints and the frontend pages.
- Implementing a search engine for fetching articles (elastic search)

Good Luck.

