# Documentation Interview TASK

1. Starting the application

To approach the task given React have been used to build the user interface and a json-server for data mocking. To run the application its very simple, we clone the project from github repository and go inside that directory created and run the terminal with the following commands:

- npm install

- npm start

After the last command a pop-up window should appear in our browser with our application. In case that doesn't happen we type in the browser the following URL:

- http://localhost:3000/

2. API

The json object is composed of different articles. Each article has an id, title, text and image. What we focus on it's the id of the article. To access the data from the json object we use the following routes:

- http://51.91.250.84:3000/articles

- http://51.91.250.84:3000/articles/:id

Ideally would be to save those in a config file, in case the ip changes, so we don't have to change that everywhere, but for the moment we leave them as they are. Using the first route we get all the articles, where in the second we get just one article that matches our id inserted in the route. There are many ways to access all the properties but we will focus on the ones specified.

3. Implementation of React

The implementation is pretty straightforward. We have the main function that serves two paths, the first one "/" which is the home page and renders a list of articles from our API and the second "article/:id" which renders a specific article. Each component stores the state of the data we access, the main difference is that the Homepage component fetches all the data and maps through it to display a list

with **link** and the titles of the articles. When we click on an article we render the second component and its matching content by using *props.match.params.id* to see the id of article that was clicked.

4. Responsive app and styling

Making the app responsive, bootstrap have been used along its classes. To obtain the break point at 768 we specify the @media screen at (max-width: 768px) for mobile and implement CSS to different classes and @media screen (min-width: 769px) for the rest of devices.

5. Functionalities and future implementations

The UI its simple to use. A list of articles appears, you click on them and takes you to the article clicked. In the article page we can refresh the page and content remains, we can go back, and access another article and repeat the process.

For future implementations, depending on the requirements, we could easily add other components or just add another features on the existing ones.

The work files inside the project are:

- App.js
- Index.html
- App.css
- /Components/Article.js
- /Components/Homepage.js