

KOJI - Technical test for back-end developers

General information

- The objective of the test is to understand your technical level, your ability to explain concepts and to resonate.
- This test is to be done independently and should not include copy pasted elements from any source. (aka Github or StackOverflow)
- You can make assumptions in order to complete the test. Don't forget to include them in your report.
- You can ask questions if you feel you are missing information.
- The test should take no longer than 1 hour. Send your progress once this time is up.
- Your code must be available on a public access Git
- You must send the answer to the questions and the repository link by email.

A. Statement:

You are in the shoes of a developer working on a new project for a business application for publishing content to print.

You are responsible for the front-end of the application. It was decided that the application would be in Typescript with React.

During the scoping meeting you got the following information:


- The application must have an authentication system with a simple role management (user, admin)
- The administrator must be able to add a certain number of sample documents with the necessary data for the generation. These documents are considered as "templates".
- A user can only access his generated documents
- A user must be able to generate a document from a "template" created by an administrator by providing the required fields.

Reflection questions:

You must answer the following questions (each question expects an answer followed by an explanation):

- What database(s) will you use? And why ?
- What libraries will you use to communicate your database? What are the necessary routes?
- How will you manage the validation of the content of the queries?
- How will you manage the pagination of the queries ?
- How will you manage authentication ?
- Bonus: How will you manage access permissions?

B. Practical case:

 This part is independent, you may choose to take a different approach than what you described in the first part.

You need to set up an api with the Javascript/Typescript framework of your choice allowing the transformation of a Markdown content into a PDF file.

- Your API can contain as many routes as necessary.
- You can approach the problem as you like as long as it is possible via one call to transform a Markdown file into a PDF file.
- The result must absolutely be a single PDF file.