Mandatory Exercise 1 - Web development in CMS

The deadline for this exercise is Friday, September 6, 23:59.

For this **mandatory exercise** you should work on **master branch only**.

Preparation

1. Create a new repository on GitHub called **mandatory-cms1**
2. Follow the instructions that GitHub gives you; Create a local repository and add a remote or clone the newly created repository.

Submission

When you submit the exercise in PingPong, before the deadline, you will enter a link to your repository, such as:

https://github.com/mygithubusername/mandatory-cms1

The teacher will look in the **master branch**. If any commits are done to the branch after the deadline, the teacher will look at the last commit before the deadline.

The repo must also contain the Cockpit database you have used. Please also include instructions on how to run the code in case it is not obvious.

You will get one of the grades **G** or **IG**.

Instructions

In this exercise you will create a simple blog application using a headless CMS called Cockpit. The application you build should be an SPA. You are free to use Angular, React or Vue when writing the application.

Cockpit

Cockpit is a very simple headless CMS that can be used to manage the content on a website. In Cockpit you can manage your content using “collections” that you specify yourself. This makes it very versatile.

You can also create “forms” and “singletons” with the CMS but in this exercise we will only use “collections”.

https://getcockpit.com/

Please use Cockpit version 0.9.2.

Cockpit API

To request data from your collections you will use the Cockpit API.

https://getcockpit.com/documentation/api/collections

The API is very simple but powerful and supports filtering, sorting and pagination.

Structure

Your application should be an SPA with at least 3 pages:

- A main page that displays a list of blog posts and links to them. You need to implement pagination on this page. The list items should contain
    - title
    - name of author
    - date
- A page that shows a single blog post. This page should contain
    - title
    - name of author
    - date
    - body of post rendered as HTML
- Authors page. This page should show a list of blog authors. Every list item should contain
    - name of author
    - description
    - photo/avatar

To be able to do this you need to create **two** collections in Cockpit: articles and authors.

An article should contain the following fields

- title - Text
- body - Markdown
- published_on - Date
- author - Collectionlink

An author should contain the following fields

- name - Text
- description - Textarea
- avatar - Image

The “Collectionlink” type makes it possible to link an article to a specific author.

The pages and fields listed above are the minimal requirements for this exercise, if you want to add additional fields or pages you are free to do so.

There are no requirements on layout and design of the application, so feel free to create your own design!

Running Cockpit with Docker

The easiest way to run Cockpit is using Docker.

If you are unable to run Docker and having trouble running Cockpit, please ask the teacher for assistance.

To start Cockpit version 0.9.2, please run the following command

```
docker run -d --name cockpit -p 8080:80 agentejo/cockpit:0.9.2
```

This will download cockpit and start it. Cockpit does not require an external database, instead it will use SQLite (a minimal database) if no external database is defined. For this exercise SQLite is sufficient.

When you are finished, please copy the “storage” folder from Cockpit, compress it and include it in your repo.

If you are using Docker, you can copy the folder with the following command.

```
docker cp cockpit:/var/www/html/storage .
```

This will copy the folder to the current directory.

Requirements

The application has the following requirements

- It should be an SPA created using Vue, React or Angular.
- It should implement correct routing
- It should contain at least three pages: a main page, pages for single blog posts and a list of authors
- It should use pagination on the main page
- It should use Cockpit as a backend and fetch data using the Cockpit API
- The body of the blog posts should be written in Markdown and converted to HTML by the client
- You should use the **Collectionlink** type in Cockpit to link authors to articles

Tips

- Start by making yourself familiar with the Cockpit interface and its API
- Try writing a simpler application with Cockpit first
- Ask questions if there is something you are unsure about