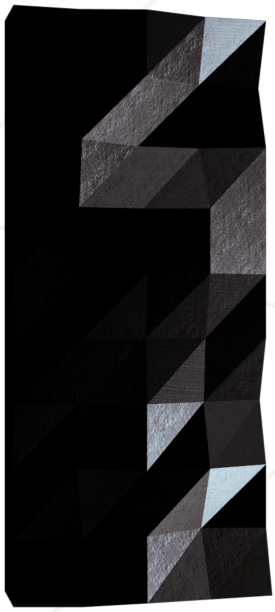

ft_pastebin

Rush Web

Summary: Ctrl-A Ctrl-C Ctrl-V

Version: 1

Contents

Chapter I

Preamble

A `pastebin`, also know as nopaste, is a web application that allow users to upload chunk of text, usually snippets of source code, for a public display.

They are very popular on `IRC` channels where copy a significant amount of text, including split into differents successive lines, is considered as not conforming to the label.

There are many similar web applications, known as “paste sites,” that have developed since the original Pastebin was launched in 2002.

Paste sites are commonly used for sharing code. However, any data in text form can be uploaded and shared. Users can use the Pastebin search tool to find relevant content based on keyword.

Chapter II

Introduction

As you may have guess, you'll need to rewrite a pastebin website. Here's a rush that looks cool you will say, my friends. web... &%@#$ WEB !.

But since we're at 42, we never do things by halves. So, for the weekend, we're going to put you on a test on a FullStack rush...

You will need to design an entire `app`, from the index.html to the build in production. And for this, we have prepared something for you to take care during the weekend!.

Chapter III

General rules

- You can use any `language`.
- You can use any local database (no Firebase for example).
- You can use any `librairies` and `framework` (except for libraries entirely recoding a nopaste...).
- The user should encounter no unhandled errors and no warnings when browsing the website.
- Everything has to be launch by a single call to: `docker-compose up -build`.

Since you will be in pairs for the next 48 hours, we advise you to distribute the tasks well between **Front-end** and **Back-end** and clearly define the technologies you want to use.

In case of doubt, you can always take PasteBin as reference.

Chapter IV

Project instructions

IV.1 Common instructions

- `ft_pastebin` consist of two separate endpoint, a front-end and an a back-end.
- The front-end part will be used to save text or code ("paste") in the database with a constraint of time and/or access. It then offer a small and unique URL to access the saved paste (ex: www.ft-pastebin.fr/ETgd475x).
- The back-end will manipulate everything related to the database. The front-end cannot change the data in any ways other than passing through the back-end.
- There should be no error, warning or logs in the front-end part. In contrast, logs in the backend are advise.
- A minimum effort will be ask on the design of the front-end. Understood that you have the choice of librairies, use that.
- The entire project must be launch with a single `docker-compose --build`.

IV.2 Front-End

Client part

The following 2 routes are mandatory:

Vue pastebin - (Route: `'/'`)

- You must propose a form that contains at least:
  - A text-area to redact a "paste".
  - A input for the title of the "paste".
  - A select with consistent duration for the "paste" expiration.
  - A select with the number of limited access.
  - A checkbox private / public.
- Upon validation, you're redirected to the newly created paste.
- A list of the latest published public "paste" (limited to 5).
- The form error must be handled before sending the POST request to the back-end.

Vue display - (Route : `'/:id'`)

- The "paste" must be shown in a specific area with his number of lines. This area cannot be modified.
- The title of the paste must be displayed as well as the created date and the time since the "paste" has been created.
- three options are available:
  - Download the "paste"
  - Display only the "paste" in a row format (without html).
  - Copy the "paste" to the clipboard.
- All informations are retrieve by GET request to the back-end.
- AN ENORNEMOUS MESSAGE if the duration or the number of access is exceeded.

IV.3 Back-end

For the back-end, you needs to handle a lot of things.

- You'll handle the table as you want, but try to be coherent.
- **Route: `'/paste/:id'`, method: `GET`**
  - Return all the informations about the "paste" according to the `id`.
  - In case of error (invalid id, limited access, duration exceeded ...), return an error message of your choice.
- **Route: `'/paste/create'`, method: `POST`**
  - Save the form informations in to the databases.
  - In case of error return an error message of your choice.
- All routes must return the correct `HTTP` code in case of success, error or unauthorized method.
- In addition to this, you will need to implement a solution that will check EVERY MINUTES the database and thus will invalidate any "paste" whom duration is incorrect.

Chapter V

Bonus part

You will get some extra points if you can:

- An options to choose syntax highlighting when pasting code.
- An admin page with the following features:
  - The page is protected by a password.
  - Show all pastes and their title.
  - Possibility to edit the pastebin of your choice
  - Delete or make unavailable a paste