T7 - Web Development

T-WEB-700

The Count of Money

specifications

0.3

Requirements

Users management

You must manage user sessions by setting up a dedicated middleware. Below you will find a list of routes that you must, *at least*, display:

- **POST** /users/register
  The user MUST NOT be logged on. Register a user by sending a form.
- **POST** /users/login
  The user MUST NOT be logged on. Simple authentication by username/password, if successful, a session is started.
- **GET** /users/auth/{***provider***}
  provider: facebook, twitter, google, etc. **The user MUST NOT be logged on**. Oauth2 authentication.
- **GET** /users/auth/{***provider***}/callback
  provider: facebook, twitter, google, etc. **The user MUST NOT be logged on**. On this route we can retrieve user's information transmitted by the third party service (or an error issued by it). This is where your API validates (or not) the user's authentication and therefore starts (or doesn't start) a session.
- **POST** /users/logout
  The user MUST be logged on. The user disconnects, so you must end your session
- **GET** /users/profile
  The user MUST be logged on. Retrieving profile information.
- **PUT** /users/profile
  The user MUST be logged on. Update profile information, which is *at least*:
  -> a nickname (their default email)
  -> a default currency for the price of crypto-currencies (default: EUR)
  -> a list of cryto-currencies, to appear on their home page
  -> a list of keywords, for their press review

CRYPTO-CURRENCIES

By default, exchange rate is displayed in EUR. The mandatory routes are, *at least*, the following:

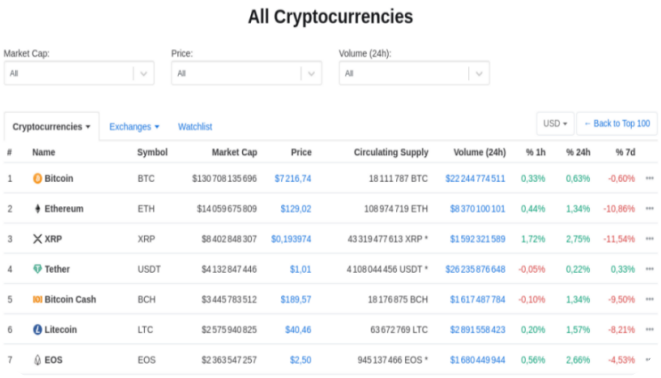
- **GET** /cryptos[?cmids={***cm***}]
  cmids: cryptocurrencies' Ids. **User MAY be logged in OR NOT**. Get the list of crypto-currencies and their info, which is *at least*:
  -> full name of the cryptocurrency.
  -> current price
  -> opening price
  -> lowest price of the day
  -> highest price of the day
  -> URL of the corresponding image of the cryptocurrency

- **GET** /cryptos/{***cmid***}
  cmid: cryptocurrency Id. **User MUST be logged in**. Returns information about a cryptocurrency.

- **GET** /cryptos/{***cmid***}/history/{***period***}
  cmid: cryptocurrency Id. ***period***: daily, hourly or minute. **User MUST be logged in**. Provides the price history of a cryptocurrency. For each period:
  -> opening, highest, lowest and closing exchange rates
  Depending on the periods, the history is shorter or longer
  -> ***daily***: Last 60 days, so 60 periods a day
  -> ***hourly***: 48 last hours, so 48 periods of one hour
  -> ***minute***: last 2 hours, so 60 periods of one minute

- **POST** /cryptos
  User MUST be logged in as well as the ADMINISTRATOR. Add a cryptocurrency to your platform. A form must be attached to the request and contain at least the cryptocurrency code, their full name and a URL for the image to which it represents.

- **DELETE** /cryptos/{***cmid***}
  cmid: cryptocurrency Id. **User MUST be logged in as well as the ADMINISTRATOR**. Deletes a cryptocurrency *(meaning that your platform does not know this currency anymore.)*

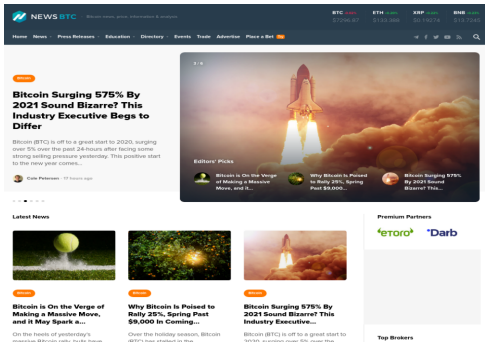


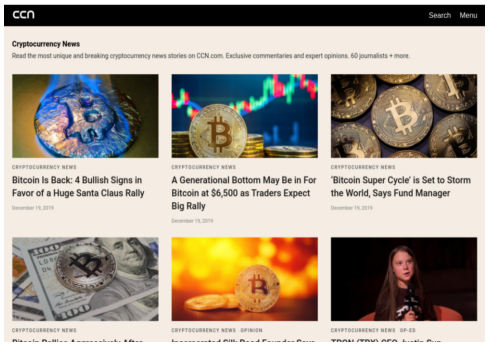
PRESS REVIEW

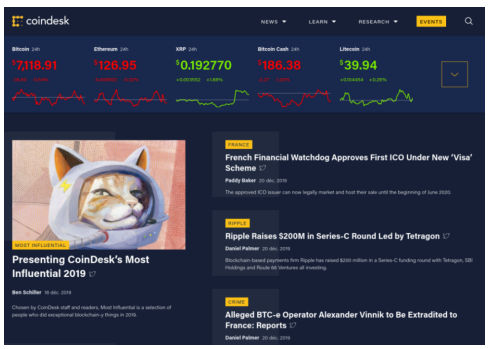
The routes you must expose, *at least*, are:

- **GET** /articles[***?params1=value1&...***]
  params: free. **User MUST be logged in (OR NOT)**. If the user is anonymous the settings (if any) are ignored and the last published articles are returned. If the user is logged in the settings are used to return only the items most relevant to the user (*a list of keywords might help you*). You are free to define the parameters that you think will be useful depending on the search options you offer to your users. Here for each article, you must provide *at least*:
  -> an id
  -> a title
  -> an URL of the article's page
  -> an URL of its image (if it exists)

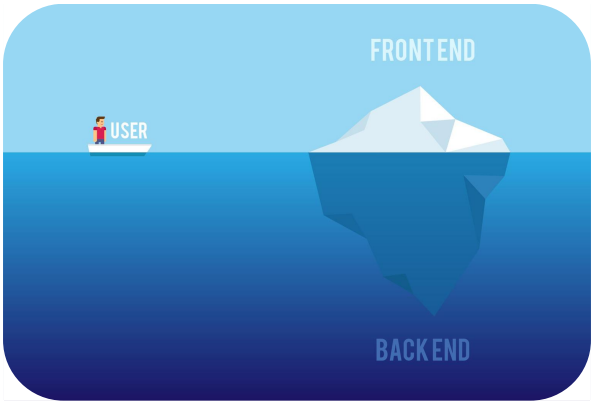
- **GET** /articles/{***id***}
  id: the Id of an article. **The user MUST be logged in (OR NOT)**. Returns information about an article, which is *at least*:
  -> the article Id
  -> the title
  -> its summary
  -> its source
  -> its date
  -> the URL of the article's page
  -> the URL of its image (if it exists)







Constraints



Containerization and orchestration

We require you to containerize each of the services with **docker** and orchestrate them with **docker-compose**.
The construction of your web application must be automated and run within the container.
Once your application built and packaged, it is served by your web server.

Code organization

The following specifications are mandatory:

- a *docker-compose.yml* file at the root of your repository
- one directory for each service
- one or more configuration files
- a *README* file abstracting your project and its deployment.

Data base

The management of your database(s) isn't subject to any constraints. Choices must be dictated by usage !

Languages and libraries

Backend must be a **node** application based on **express.js** framework. Frontend must be in **React OR Angular OR VueJS**. Feel free about your frontend's design. Be creative but do not neglect user experience. Distinguish the good, the bad, and the ugly!