T7 - MSc Pool

T-POO-700

Web Interfaces

Project

1.3.3

EPITECH.

Web Interfaces

delivery method: Github
language: JS

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

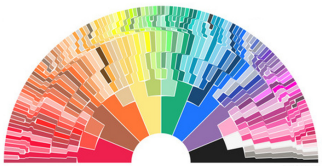
Before you start, make sure you have **finished** and **assimilated all the concepts** discussed in the Bootstrap. In addition, you will need to use the API that you developed previously, so make sure it is functional.

A meeting is planned between your manager and the Mayor of Gotham next week.

Your manager must make a first demo of the application. He asks you to put in place the employee information display so that he can give an overview of the final result.

Create a user interface that displays graphs and dashboards to visualize a person's working time.

This project is part of a problematic of *Data Visualization*, which consists in the graphic representation of figures or raw data.



Some organizational constraints have also been imposed by your project manager :

- you must use the JavaScript framework Vue.JS to create the interface.
- you can use any tool you fancy for graphics, we advise you `vue-charts` but your free to do as you want.
- for ergonomics reasons, you will only need **one and only one** view, defined in a `App.vue` file in the`/ src` folder.
- all the components must be in the `/src/components` folder.
  Five components are required :

  1. **User**
     This component will be used to identify the current user, and must be present on all pages of your web application.
     It must implement the following methods (with self-explanatory names) : `createUser()`, `updateUser()`, `getUser()` and `deleteUser()`.

  2. **WorkingTimes**
     This component will be used to display the working times recorded by the API. It is connected to the */workingTimes/:userI* route.
     It should also have at least the `userId` and`workingTimes` data (the table summarizing the offset times) and the `getWorkingTimes()` method.

  3. **WorkingTime**
     This component will be used for displaying, creating, modifying and deleting a working time.
     It will be linked to the routes */workingTime/:userid* (for creation) and */workingTime/:userid/:workingtimeid* (for modification and deletion).
     It will implement the methods `createWorkingTime()`, `updateWorkingTime()`, `deleteWorkingTime()`.

  4. **ClockManager**
     Connected to the */clock/:username* route, this component will be used to declare hours worked.
     It must have `startDateTime` data (is worth *null* if no work period is in progress) and `clockIn` (a boolean that is *true* if a work period is in progress) and the `refresh()` methods and `clock()` (to pass from active to inactive and vice versa).

  5. **ChartManager**
     Connected to the */chartManager/:userid* route, this component is used to manage the graphs.

  You can create a specific component per chart, but you **must** have the ChartManager component. If you decide to use different components for each chart, you **must** use the Vue.JS routing system.

- you must come up with at least three graphs, configurable, and of different types (bar, line, pie, radar or other type of chart) ; your manager would like to impress the mayor, who is used to cryptographic sequencers and other bat gadgets.

ALL your dates and times should be stored as follows : “YYYY-MM-DD hh:mm:ss”.
Look at the *watch* side of the components.

The overall rendering of your application and its ergonomics are essential for a good user experience. A good UX and appropriate features must be your priority, otherwise your application may never be used!