Technical Documentation

Getting Started

A brief introduction

Dashboard is a browser integration project realisated by two students. The objective was to create a dashboard where you can apply and manage different widgets.

This project is compliant with the MVC architecture.

The functionalities

We can split the application in three parts:

- The apply panel

For the apply panel, the user has to define what widgets he wants to apply by clicking on buttons that represent these widgets. For the ones who require Oauth2 authorization, they have to log into the platform by redirection. The application gets an access token and the user can use it for several requests.

- The widget applied panel

In the widget applied panel part of the application, the user can see the widgets he has applied for and manage them.

- The profile

The user can access to some simple data from his profile and can logout.

The technologies used

The project was developed on phpStorm in Laravel(Php and React.js). The website runs thanks to Docker.

In order to implement a new widget/service :

Back-End : you will have to make a new Controller in app/Http/Controllers/ and link it to the internal API by creating routes used for the front.

Front-End : you will have to create a new Component in resources/js/components/ and create a function that will render in HTML/CSS what you want it to display.

API references

Tons of api were used

The Coinbase api.
The Weather Prediction api.
The Gmail api.
The Spotify api.
The Youtube api.
The Love Meter api.
The Covid Infos api.
The Google Calendar api.

Routes references

/api/coinbase/auth

Used to call “getAuthorisationCode()”, it generates the link to the Coinbase oauth section.

/api/coinbase/token

Used to call “generateAccessAndRefreshToken()”, it gets the code returned by the Coinbase oauth section thanks to the first route call.

/api/coinbase/token/refresh

Used to call “refreshAccessToken()”, it prompts the Coinbase oauth section for a new access token.

/api/coinbase/data

Used to call “getData()”, Api call to get user data.

/api/coinbase/crypto/{currency}/{exchangeCurrency}

Used to call “getCurrencyData()”, Api call to get currency value data.

/api/gmail/auth

Used to call “getAuthorisationCode()”, it generates the link to the gmail oauth section.

/api/gmail/token

Used to call “generateAccessAndRefreshToken()”, it gets the code returned by the gmail oauth section thanks to the first route call.

/api/gmail/token/refresh

Used to call “refreshAccessToken()”, it prompts the gmail oauth section for a new access token.

/api/gmail/data

Used to call “getData()”, Api call to get user data.

/api/calendar/auth

Used to call “getAuthorisationCode()”, it generates the link to the calendar oauth section.

/api/calendar/token

Used to call “generateAccessAndRefreshToken()”, it gets the code returned by the calendar oauth section thanks to the first route call.

/api/calendar/token/refresh

Used to call “refreshAccessToken()”, it prompts the calendar oauth section for a new access token.

/api/calendar/data

Used to call “getData()”, Api call to get user data.

/api/spotify/auth

Used to call “getAuthorisationCode()”, it generates the link to the Spotify oauth section.

/api/spotify/token

Used to call “generateAccessAndRefreshToken()”, it gets the code returned by the Spotify oauth section thanks to the first route call.

/api/spotify/token/refresh

Used to call “refreshAccessToken()”, it prompts the Spotify oauth section for a new access token.

/api/spotify/data

Used to call “getData()”, Api call to get user data.

/api/spotify/data/{user}

Used to call “getSpecificUserData()”, Api call to get specific user data.

/db/insert/{widget_id}

Used to call “add_row_in_widget_order_table()”, Stock the order of the widgets applied.

/db/retrieve/

Used to call “get_widget_order()”, Retrieve widget applied order of display.

/api/weather/{city}

Used to call “create()”, Api call to get data.

/api/love/{fperson}/{sperson}

Used to call “create()”, Api call to get data.

/api/youtube/{channelName}

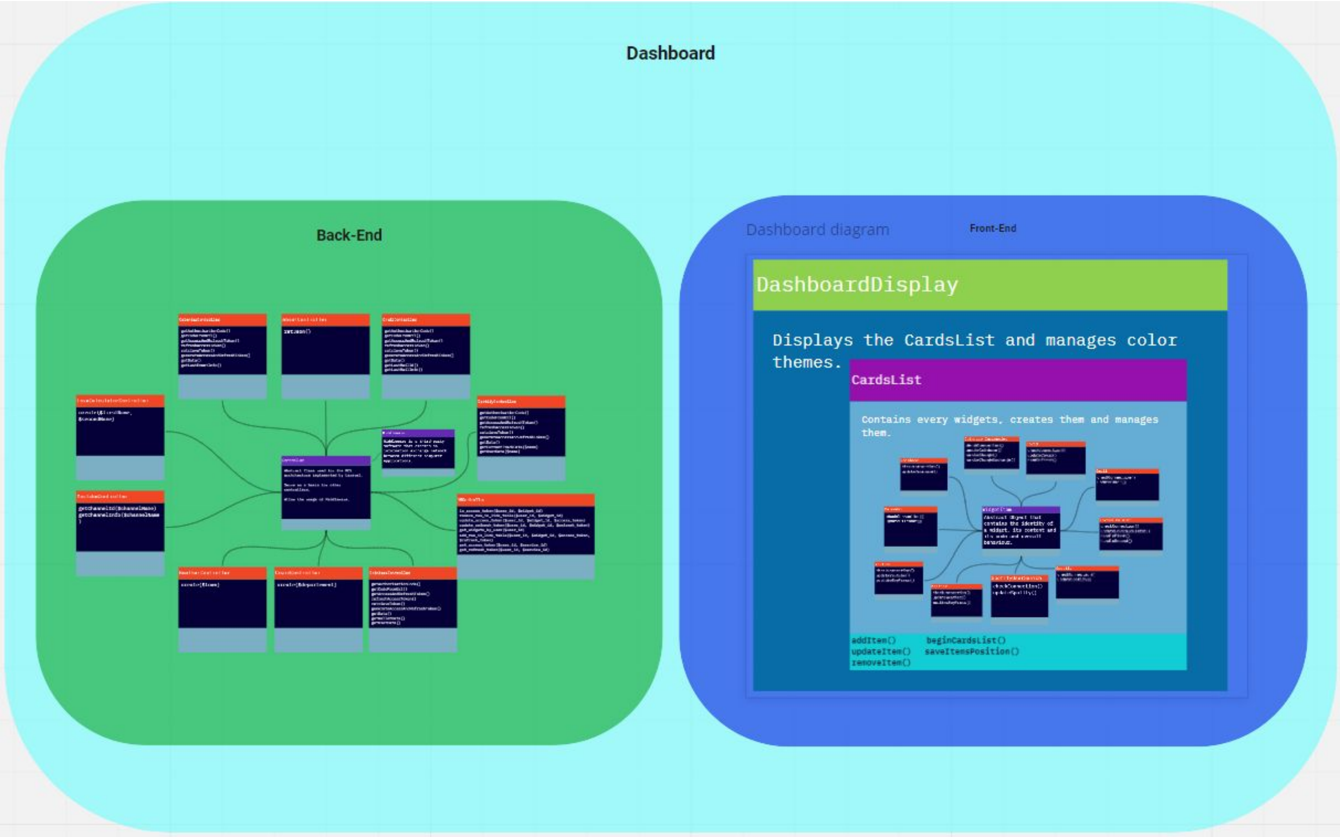
Used to call “create()”, Api call to get data.

/about.json

Used to call “retJson()”, return the widget architecture in json format.

React components

UML Class Diagram



Documentation available at path ***https://miro.com/app/board/o9J_ldM85JY=/?utm_source=notification&utm_medium=email&utm_campaign=daily-updates&utm_content=go-to-board***.

Documentation realised by Kevin Pruvost & Florian Lamant