Cryptocurrency price monitoring web application design

Team : 640

Yuqing Cheng, Peng Tong, Xue Zhao, Yuan Gao

1 INTRODUCTION AND APP DESCRIPTION

2 INTERACT WITH APPLICATION

There are four main pages in our application.

2.1 Index Page

Index page contains three main parts: navbar, news, and coin info of three types of cryptocurrency.

- If the user is not logged in, he/she can only see the log in and sign up dropdown on the navbar.
- By clicking the log in dropdown, the user can log in with google or github account. Or he/she can log in using his/her own account on our website.
- By clicking the sign up dropdown, the user can sign up an account on our website.
- If the user is logged in, he/she can access coins info page, alert page and notification center page via button on navbar.
- Left part of main body in index is news part, where our application will show the latest cryptocurrency news. By clicking the news title, we can view the detail of that new on news websites.
- Right part of main body in index is coin info part, where our application will show the real time price, change since open, and open price of three different coins(BTC, ETH, LTC). By clicking the coin name, we can enter the specific info page of that type of coin.

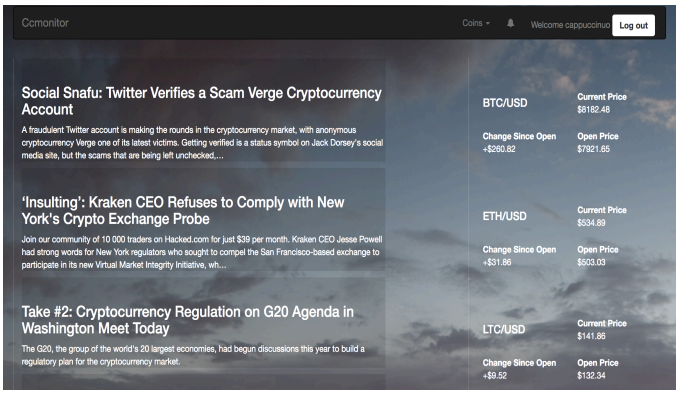

Figure 1: Index Page

2.2 Coin Info Page

The coin page show specific info of each coin. On the top of the page, we display the same brief information on index of that coin. Right of that part we use a chart to display the trend of that coin, including real time price trend, day price trend, week price trend and month price trend. On the bottom part of the page, we display a table that shows 30 days historical price of that coin.

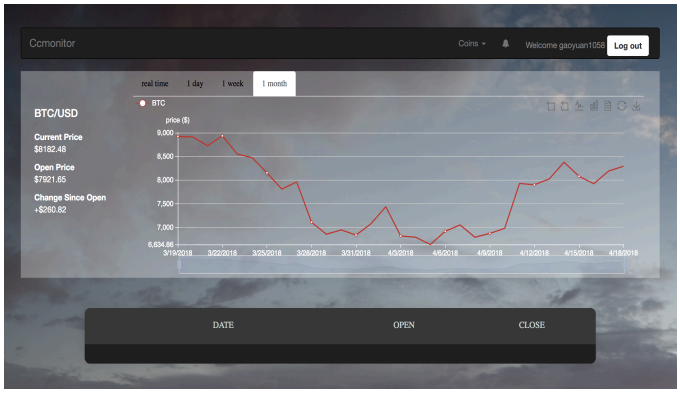

Figure 2: BitCoin Info Page

2.3 Alert Page

There are two parts of our alert page, new subscribe and my alert.

To subsribe a new alert, the user can click coins on nav bar and click subsribe. Users can choose coin type(BTC, LTC, ETH), alert type(Ascending or Descending), as well as the threshold of that alert.

Figure 3: Alert Page

To check all the alerts that the user have, the user can click my alert on alert page, the user can delete the alert by clicking the trash icon.

2.4 Message Page

The user can check all the messages the server sent him/her by clicking my alert on alert page. Also the message page provide the function to filter all the message, including filter the messages by coin type, filter the messages by alert type.

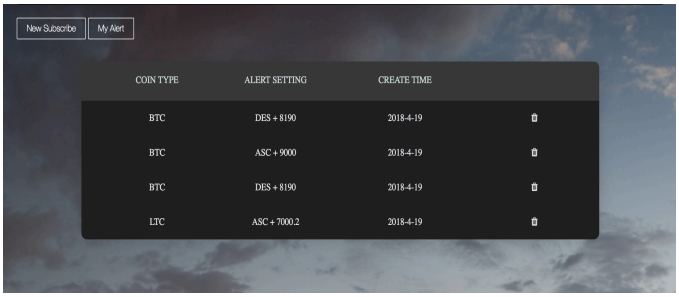

Figure 4: `My Alert Page`

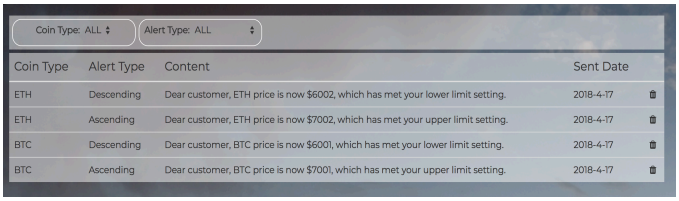

Figure 5: `Message Page`

3 SERVER-SIDE STATE

3.1 Database Entities

We store some entities permanently in postgres db at server-side. These entities include person, alert-setting and messages. A person has an id, name, email, hashed_password, provider (google or github) and a token from provider. An alert-setting has an id, threshold, coin_type (BTC, ETH or LTC), alert_type (ascending or discending) and a person_id as foreign key. A message has an id, content, coin_type and alert_type used for filtering at front-end. Browser can access and update/delete these entities through corresponding APIs and controllers using ajax call, and ajax callback would dispatch the front-end store and reflect changes.

3.2 Agent State

Apart from those in-database entities, real-time prices of cryptocurrencies are stored in Phoenix Agent and pushed to browser through web-socket by Phoenix Channel. The real-time price data is fetched from coinbase API through our server-side Nodejs client and supplied to Phoenix Agent through web-socket, too. In the Phoenix Agent, the state of real-time prices has a structure like this:

%"BTC" => [], "ETH" => [], "LTC" => [], "time" => [],

where a series of prices in last 5000 seconds with 10 seconds as interval for each cryptocurrency is stored, also the corresponding timestamps are stored. The series of prices and timestamps have a maximum length of 500, which is maintained by the Nodejs client. These real-time prices are used to support the front-end echarts for users to observe a real-time price change in 5000 seconds.

3.3 Structures in Store

We maintained a store to create and manage a set of data structures, utilizing Redux to combine these structures into a completed state for components to render. Data structures in store include: "users" that contains all users in our application, "token" contains the token of current user, "login" contains login information, "signup" contains register information, "alerts" contains all the alerts set by current user, "messages" contains all the emails that have been sent to current user, "current_coin_type" represents which type of coin user is viewing, "prices" contains the real time prices for coins, "historical_prices" contains historical prices for coins.

4 RELATED API

4.1 Coinbase

We used Coinbase API to get real time prices for coins. Real time price provides important information for users to keep track of the newest status of interested coins. User behaviors like purchase and sell are closely related to real time price. Besides, real time price is the foundation of other functionalities of our application, like sending messages when the price reached certain threshold configured by users. Coinbase provides simple API to get real time price of different coin types. On the server side, we got real time price every five seconds, then pushed price into browser-side state through web socket.

4.2 Crypto Compare

We used CryptoCompare API to get historical prices for coins. Historical price data can provide users a straightforward sense of the trending and potential of coins, which is necessary for users to make trading decision. And CryptoCompare provides free and concise API to get historical price data. It parameters to set the time period to aggregate the data over (for daily it's days, for hourly it's hours) and the number of data points to return. We used CryptoCompare to get the past 24 hours, one week, one month, six months and one year's history prices for coins, saving these prices into state and rending them into charts and tables.

4.3 Crypto Coins News API

We use crypto coins news api, which provides breaking cryptocurrency news - focusing on Bitcoin, Ethereum, ICOs, blockchain technology, and smart contracts. We can fetch all the news in JSON format by using AJAX's GET method. To display the news, we use title, description, and URL of that JSON data. Every time the user enter our page, the page will display the top 5 crypto news. In this way, the user can know the latest news of cryptocurrency.

4.4 OAuth API(Google and Github)

5 COMPLEX PART OF APPLICATION

5.1 Real Time Prices Monitoring at Server Side

A complex part of our app is finding a way to monitor real-time prices for cryptocurrencies at server-side. This is necessary because our app needs to:

- 1. reflect real-time price change through e-charts,
- 2. send user alert emails based on the real-time price change.

Actually, it is the second point makes it necessary to monitor real-time prices in server instead of in browser. However, we used coinbase as external API to get real-time price data, but it doesn't supply a Phoenix/Elixir SDK, so we need to find a work-around.

Our solution is building a server-side Nodejs client, which could work with a corresponding SDK from coinbase and connect with Phoenix server through websocket. In the Nodejs client, we used a recursive setTimeout function to fetch real-time price data every 10 seconds, and we used Phoenix Channel to push the data to Phoenix server through websocket. Since the real-time browser data is also supplied through websocket, it's convenient to let Nodejs client and browsers join at the same channel, so every price update in Nodejs client could reflect in browsers through this channel. Since the Nodejs client is deployed on server and is the only data source to update price data in channel, thus the monitoring of real-time price data on server is implemented.

5.2 Display Data via Chart

To give users a clear and straightforward presentation of coin prices, in PriceChartComponent, we utilized Echarts to draw the chart for real time prices and historical prices of selected coin. While implementing this component, first we need to distinguish real time price and historical price, because real time price is pushed by server through socket and history price is returned by API request. To achieve that, we maintained a variable to indicate whether user wanted to view real time price chart or historical chart, detecting whether the variable need to be toggled after each user selection. Besides, since we provided three different coin types, the chart should have ability to show price of different coin types in a single component. To achieve that, we maintained a current_coin_type variable in state to indicate current coin type being selected. Based on the value of this variable, chart will show corresponding prices for specific coin type. Also, we need to adjust the apperance of the chart (like font, color, axis) to keep consistency with the overall page style.

6 CHALLENGES AND SOLUTIONS

6.1 Authentication with Social Media Account

A challenge we faced was implementing login with Github/Google in our SPA. The auth controller in Phoenix works well with html.eex templates by putting call-back user information in session. However our app is a SPA that maintains a browser-side state as store, which is not easy to get user information from session. Besides, our app supports signup/signin with out website account instead of Github or Google, it is uneasy to integrate three type of users together and let them behave equivalenty.

Our solution is giving every user a server-generated token with Token.sign and use this token to uniform users from different login providers. For our own users, the token is generated at TokenController as soon as they successfully login, and is sent back to browser through ajax callback. As for user login by providers, their user_id is first fetched at PageController from session, and the token is generated at PageController and sent to browsers through window variable. Furthermore, the user information in session will be removed at PageController to avoid confusion in the future. Our front-end app component will mount to detect token from cookies and token from window variable to update the login status.

6.2 Alert Email Distribution