Mister-BITCoin

Building it with Angular!

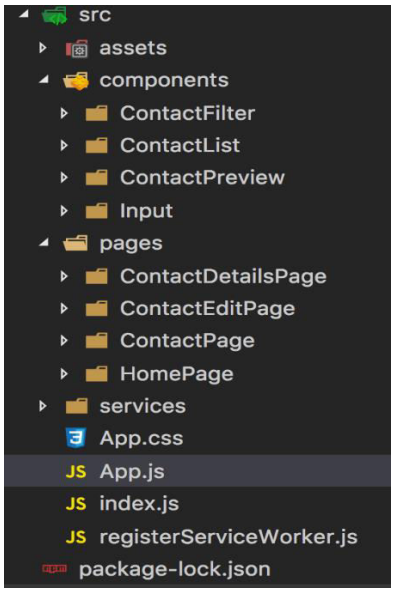
Let’s build a digital wallet for holding bitcoins and sending (paying) them to contacts.

Note that this app is frontend only and the money goes from the user balance to his contacts (not to real users)

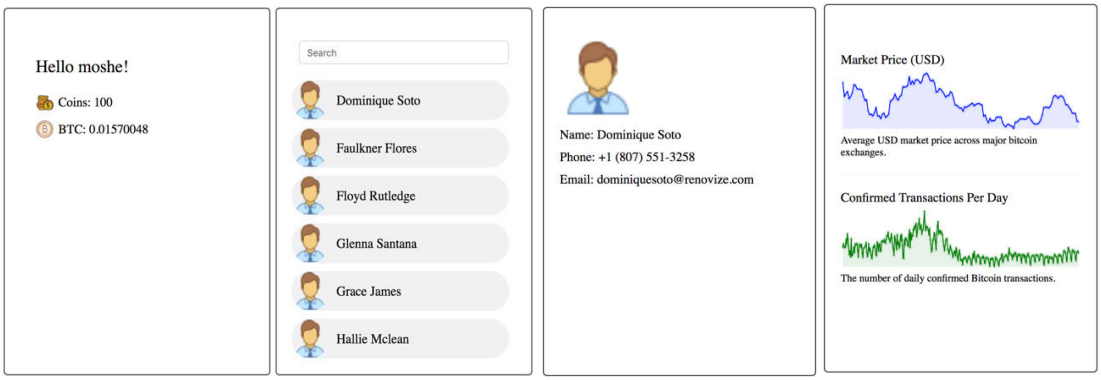
Start by creating the following pages (We will later have full routing, for now you can comment-out components or add some buttons to switch between "pages")

Directory structure

Please follow the following structure and naming conventions:



Part 1 Contacts



Services

ContactService

Use the provided ContactService

Example to contact model:

```
{
  "_id": "5a56640269f443a5d64b32ca",
  "name": "Ochoa Hyde",
  "email": "ochoahyde@renovize.com",
  "phone": "+1 (968) 593-3824"
}
```

UserService

getUser()

this function will return a user (currently hardcoded and synchronously)

Example for user model:

```
{
  name: "Ochoa Hyde",
  coins: 100,
  moves: []
}
```

BitcoinService

You can start by using fetch or axios for getting the data, then switch to the Angular HttpClient that returns an Observable.

getRate(coins)

- Returns Promise
- Bitcoin rate (use a Bitcoin value API such as this)

getMarketPrice(), getConfirmedTransactions()

- Returns Promise
- Return chart data as described below.

Charts:

Here are some APIs to fetch data from:

1) trade-volume
    a. Site - https://www.blockchain.com/charts/trade-volume ,
    b. JSON: https://api.blockchain.info/charts/trade-volume?timespan=5months&format=json&cors=true
2) avg-block-size
    a. Site - https://www.blockchain.com/charts/avg-block-size ,
    b. JSON: https://api.blockchain.info/charts/avg-block-size?timespan=5months&format=json&cors=true
3) market-price
    a. SITE- https://www.blockchain.com/charts/market-price ,
    b. JSON: https://api.blockchain.info/charts/market-price?timespan=5months&format=json&cors=true

You can find more of the APIs here.

Note: to prevent the API blocking you for too many requests, save the response in the service (or local storage) for development and later switch to using the real API

Note: Some chart's API calls are available with CORS headers, add a &cors=true parameter to the GET request.

Note: You can add 'timespan=XXX' to fetch more/ less data
(XXX can be one of: {X}months, {X}days, {X}years)
Url query example:

```
https://api.blockchain.info/charts/market-price?timespan=5months&format=json&cors=true
```

Pages

<HomePage>

Use UserService.getUser and BitcoinService and display:

- User Name and Coins
- Current Bitcoin rate

<ContactPage>

Gets contacts from ContactService and renders a *<ContactList>* component, passing down the contacts.

<ContactDetailsPage>

Get the contact by given contactId from ContactService and render the contact details (currently get the contactId from props or hardcoded)

<StatisticPage>

Display the charts:

- Market price
- Confirmed transactions per day

You may use/ add other charts if you like

Components

<ContactPreview> Props: contact

Render a div with an image (You can use robohash) and a span for preview

<ContactList> Props: contacts

Render each contact previews inside an

<Chart>

Render a chart
Props for example -: title, data, description, color…
Use a charts library like angular-google-charts

<ContactFilter>

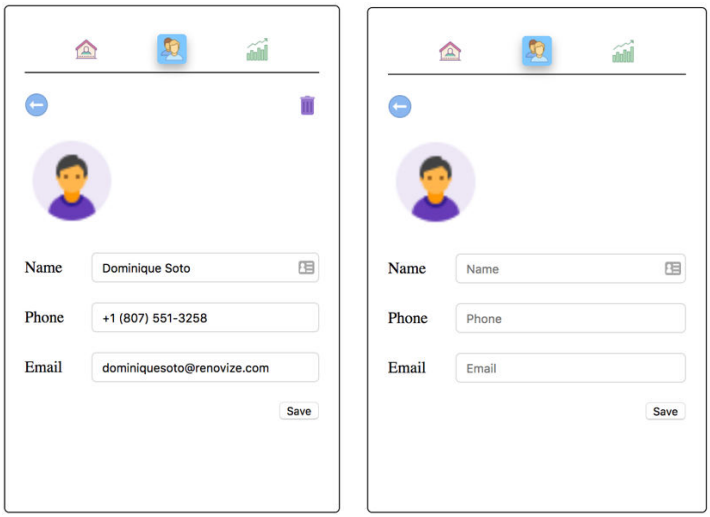
Allows free text search by name / phone and calls onFilter.emit() on every keypress (onChange), passing a filter object e.g. : {term: 'puk'}

GIT Push, Go Home.

Part 2 CRUDL

Add Router, Header and implement the full CRUDL on Contact.

Note – for routing to work in github pages, we will need to use the *HashRouter* and not the *BrowserRouter*.



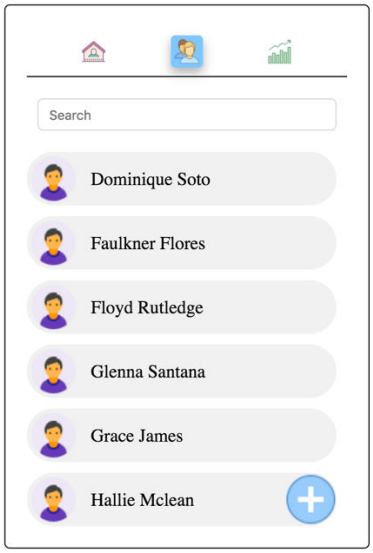
Pages

<HomePage> (route: /)

1) make sure you can access to this page from route

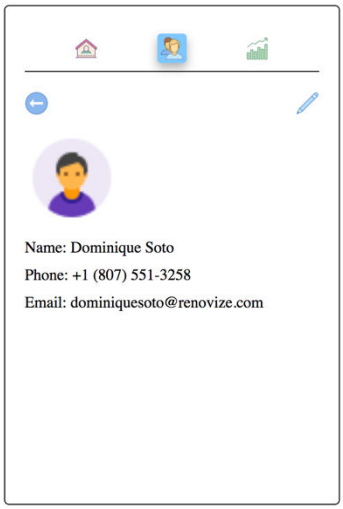
<ContactPage> (route: /contact)

2) make sure you can access to this page from route
3) add new contact button (when user click it will move to *<ContactEditPage>*)



<ContactDetailsPage> (route: /contact/:id)

1) Change the component so now you will receive an id as route param and gets a contact from the ContactService, display that contact in full.
2) Add navigation buttons:
   Back – when clicking navigate back to *<ContactPage>*
   Edit – when clicking navigate to *<ContactEditPage>*

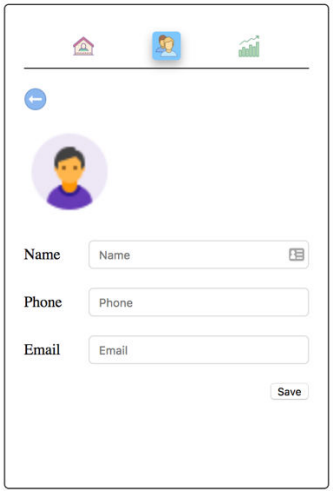


<ContactEditPage> (route: /contact/edit/:id?)

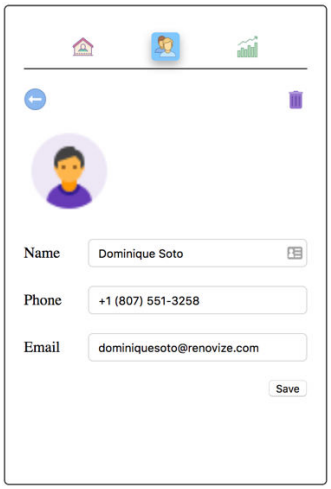
Allows Adding and Editing a contact

- Gets a contact from the service by id or start with a new contact
- Allow editing the name, email and phone of that contact

CREATE MODE:



EDIT MODE:



- Add action buttons:
  Back – back to contact details
  Delete – remove the contact and navigate to *<ContactPage>*

Components

<Header>

Render a div with a link so we can navigate between different pages

<ContactList>

Add <Link> element to add the ability navigate to contact details page when clicking on each contact

Add the project to github

Edit the manifest with colors and icons, and check your PWA from Mobile

GIT Push, Go Home.

Part 3 User authentication

Services

UserService

```
{
   name: "Ochoa Hyde",
   coins: 100,
   moves: []
}
```

Add the functions:

- signup(name)
- addMove(contact, amount)

Use the local storage to save/ load the user.

Move model:

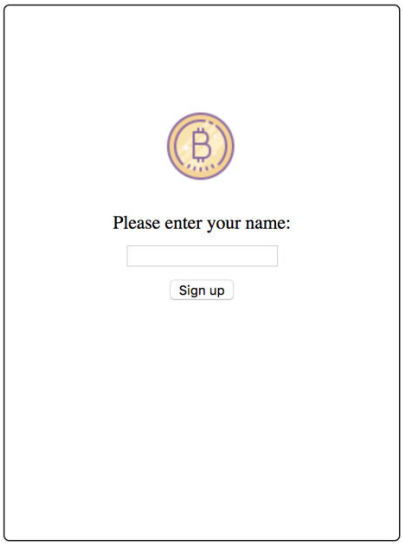
```
{
   toId: "d99e3u2ih329"
   to: "Moshiko",
   at: 2652712571,
   amount: 2
}
```

PAGES:

<SignupPage> (route: '/signup')

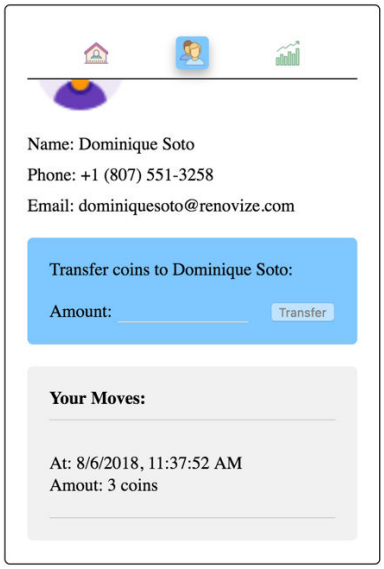
Ask for user name and save the new user in local storage and local variable using the UserService.

- When user is not known we route to this page
- The <SignupPage> just requests a name
- New user gets 100 coins when signup
- To keep it simple, do the signup process synchronously (no need for promises here in UserService)



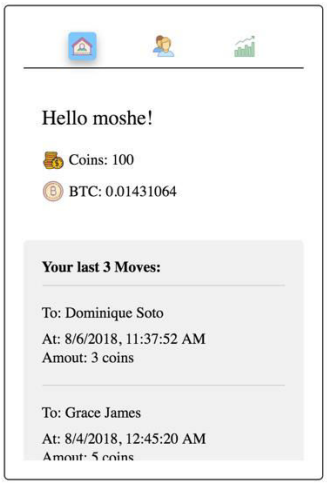
<ContactDetailsPage>

- render a *<TransferFund>* component – allow to move coins from user to this contact.
- render a *<MovesList>* component - display all moves to current contact



<HomePage>

- render a *<MovesList>* component - display the last 3 transactions



Components

< MovesList > props: title, moves-list

- display a list of moves using the UserService

< TransferFund > props: contact, maxCoins, onTransferCoins

- show a Transfer Fund form (with an amount field).
- when submitted (call to *onTransferCoins*):
  1) call to UserService to add a move.
  2) reduce from the user balance (this money goes nowhere!) using the UserService.

  Note: at this point you will need to refresh the page to see the new transaction in *<MovesList>.* you can add callback as props to render the *<ContactDetailsPage>* but when we will use the state management it will render automatically.

Some Inspiration