

Mister-BITCoin native

Building stuff with React Native!

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

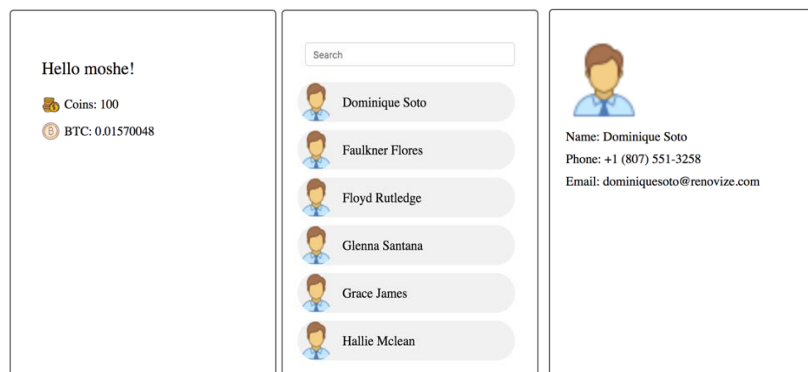
Start by creating the following screens.

As we don't have routing yet – and to keep you focused, you can comment-out components or add some buttons to switch between pages.

Directory structure

Please follow the regular structure. Use a screens folder instead of the regular pages folder.

Part 1 Contacts



Services

ContactService

Use the previous ContactService!

Example to contact model:

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

UserService

Use the same userService you had in the previous project.

BitcoinService

Use the same bitcoinService you had in the previous project

Screens

<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)

Components

<ContactPreview> Props: contact

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

<ContactList> Props: contacts

Use a `FlatList`

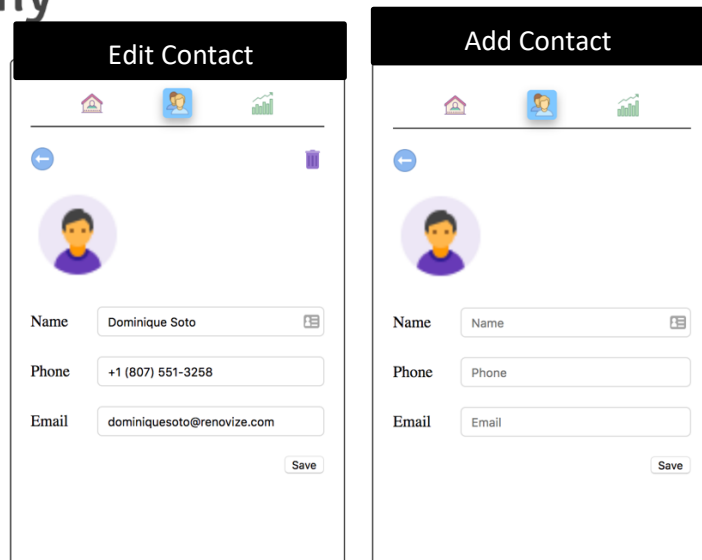
<ContactFilter> Props: onFilter

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

GIT Push, Go Home.

Part 2 CRUDL

Add Navigation, implement the full CRUDL on Contact.

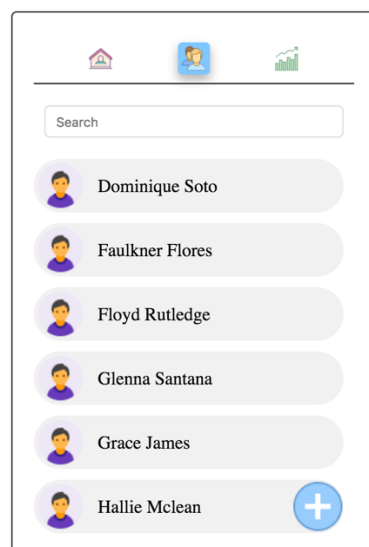


Screens

<HomeScreen>

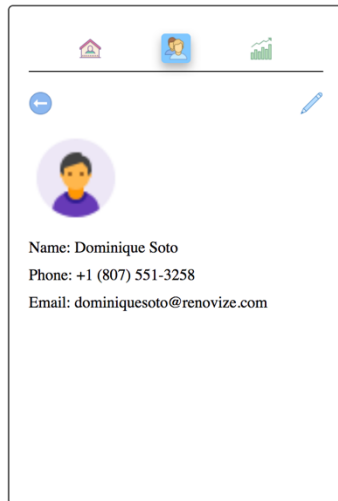
<ContactsScreen>

- 1) add new contact button (when user click it will move to *<ContactEditScreen>*)



<ContactDetailsScreen>

- 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>

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 View with Buttons and use the navigation.navigate function

<ContactList>

Add a press event to the element to add the ability navigate to contact details page when clicking on each contact

Edit the manifest with colors and icons

GIT Push, Go Home.

Part 3 User authentication

Services

UserService

Add the functions:

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

Use the local storage to save/ load the user.

Move model:

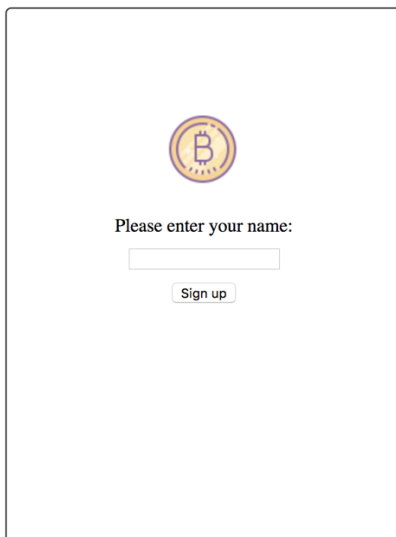
```
{
  toId: "d99e3u2ih329"
  to: "Moshiko",
  at: 2652712571,
  amount: 2
}
```

PAGES:

<SignupPage> (route: '/signup')

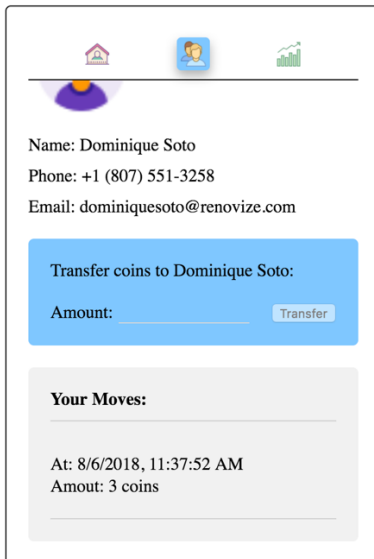
Ask for user name and save the new user in local storage and local variable using the [UserService](#).

- 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](#))

A screenshot of a web form for signing up. At the top center is a Bitcoin logo. Below it, the text "Please enter your name:" is displayed. Underneath the text is a single-line text input field. At the bottom of the form is a button labeled "Sign up".

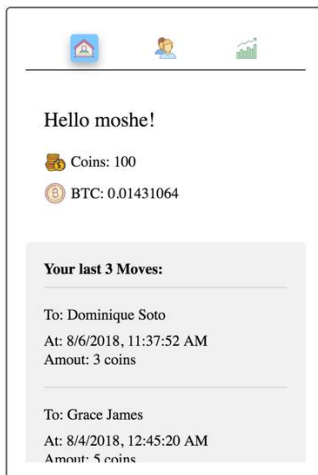
<ContactDetailsPage>

- 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.

Part 4 Getting serious - State management

Use Mobx, Add a store and manage your state like a pro

Part 5 Over the edge

1. Support offline
 - Show an Offline / Online indication (see [navigator.onLine](#))
 - Keep the BTC and last charts data in local storage
 - Use local data first, then get from network
2. Add unit testing
3. Deploy your components to storybook

Some Inspiration

