Rematch example

This example has two pages. The first page has a counter which can be incremented synchronously or asynchronously. The second page is a page which shows a list of github users. It fetches data from the github api using this endpoint.

Since rematch is utility which uses redux under the hood, some elements like `store.js` and `withRematch` are very similar to the `with-redux` example. Please go through the `with-redux` example before reading further if you are not familiar with how redux is integrated with Next.js. Rematch is just an extension for Redux so a lot of elements are the same.

Deploy your own

Deploy the example using Vercel or preview live with StackBlitz

▲ Deploy

How to use

Execute `create-next-app` with npm or Yarn to bootstrap the example::

```
npx create-next-app --example with-rematch with-rematch-app
# or
yarn create next-app --example with-rematch with-rematch-app
# or
pnpm create next-app -- --example with-rematch with-rematch-app
```

Deploy it to the cloud with Vercel (Documentation).

Directory structure

Besides the `pages` directory, there is a directory called shared which holds all of the code belonging to rematch. `Rematch` has a lot lesser boilerplate than `Redux` because it is able to put actions(including async actions), *models* and reducers together. Hence, a `models` directory is present, which contains the logic for `counter` and `github` users.

Some features of this example are :

- Pages are connected to rematch using `withRematch` util. These pages are capable of accessing values from the store and dispatching changes
- Components are inside the `shared/components` folder. The `counter-display` component is connected to the store using the `connect` function to show how components which are not pages, can connect with Rematch.
- The file `shared/store` exports an initStore function which is used by `withRematch` to create store universally on the server and on the client.