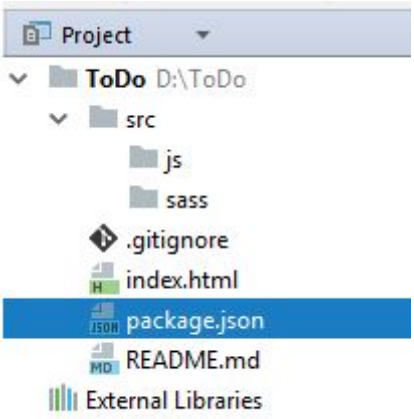
Jak zacząć projekt?

I. Konfiguracja

1. Stwórz repozytorium na github.com

2. Stwórz wstępną strukturę folderów na dysku



3. Połącz repozytorium z Twoim folderem.
```
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/[Twój-login]/ToDo-React.git
git push -u origin master
```

4. Stwórz package.json
   npm init

5. Zainstaluj npm install
   Wszystko to co jest package.json.
   https://github.com/agatablue/ToDo-React

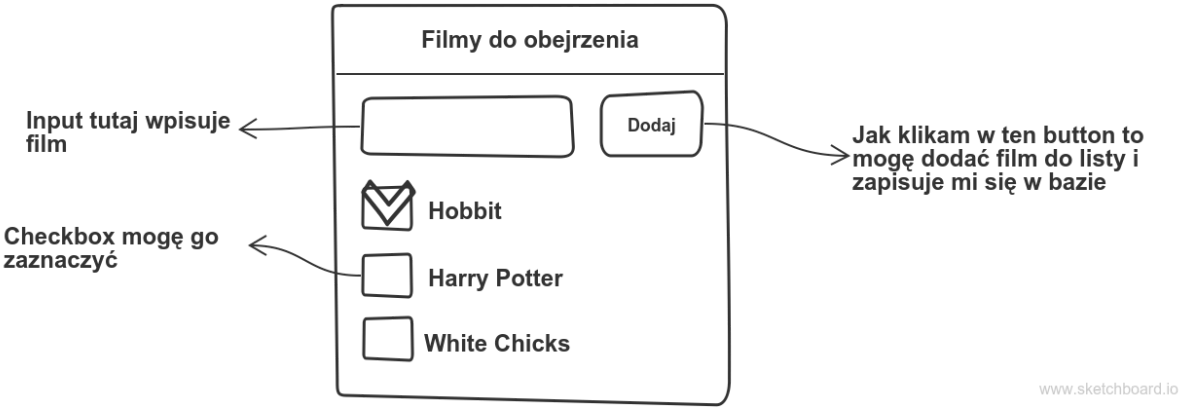
6. Ustawienie webpacka
   Skonfiuguruj Webpack w sposób, który jest w repozytorium.

7. Uruchom projekt wpisując npm start.
   Teraz możesz go otworzyć w przeglądarce http://localhost:3001/

II. Stworzenie widoków

(Commit nr ccc1a34150871338)

1. Makietę już masz prawda? Na kartce? W aplikacji? Nie ważne. Ważne, że wiesz co chcesz zrobić. Ta makieta niżej została zrobiona w https://sketchboard.me/



2. Możesz sobie również stworzyć listę zadań do zrobienia np. tutaj: https://trello.com/

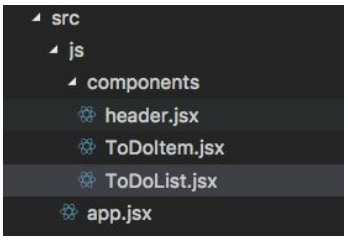
3. Teraz przygotuj html i Css - nie musi być w jsx, może być osobno. Możesz stworzyć osobny folder np. mock-ups i tam umieścić osobno tylko widok, bez funkcjonalności. Spójrz w folder repozytorium mock-ups.

4. Przenieśmy teraz widok do Reacta.
   Trzeba się tutaj zastanowić jak podzielić projekt na mniejsze komponenty.
   Pierwsza zasada - nie przesadź na początku z ilością komponentów. Na początek im mniej tym lepiej.
   Na przykładzie naszej listy zróbmy na razie jeden ToDoList.
   Widzę, że będę potrzebować trochę więcej tych komponentów.
   Zagospodarujemy zatem folder src/js

5. Tworzymy w folderze js folder components - > i każdy komponent będziemy umieszczać w osobnym pliku. Na początku jak zaczynamy dzielić to wychodzi tak:

   1. Zrobie sobie jeden wielki komponent, a w nim bedzie wszystko.
   2. Potem myslisz dobra zrobie ToDoItem - dla kazdego li i wygeneruje w ToDOList
   3. Potem myślisz sobie gdzie wrzuic ten napis “Moje ulubione filmy i ten input z przyciskiem służace do dodawania filmów i tworzysz 3 komponent header

I mamy 3 plus plik startowy app.jsx.



6. Po takim podziale mamy ustawiony widok, teraz czas na funkcjonalności.

III. Refactoring

Refactoring - to poprawianie kodu. Pierwsza zasada refactoringu mówi “ Nie rób refactoringu jeśli nie masz testów”

No ale my nie znamy jeszcze testów - więc musimy spróbować ją złamać.

(Commit nr `990dc5ea44218c769`)

1. Z makiety wiemy, że jak kliknę w przycisk dodaj to mogę dodać element do listy.
2. I tutaj możemy natknąć się problem. Otóż header.jsx ładujemy w app.jsx, a bedziemy musieli z niego przekazać informacje do ToDOList, co wydaje się beznadziejnym pomysłem, bo app.jsx mialo nam tylko startowac aplikacje a nie zawierac logikę.
   I tu przychodzi czas na PRZEPROJEKTOWANIE

   Chodzi o to, że nigdy nie będzie tak, że zrobisz aplikacje idealną od początku.
   Po drodze duże rzeczy nas spotyka i wtedy zmieniamy pierwotne koncepcje, trzeba się do tego przyzwyczaić.
   Zmiany :
   - nie mamy przeciez listy to Do tylki liste z filmami - zmienmy na movieList i movieItem - zmieniam też nazwy plików na małe llitery, żeby być konsekwętną w nazewnictwie wszystkich plików.
   - zmieńmy również nazwę app.jsx na main.jsx <- app oznacza aplikację, a main to coś głównego, app przyda się nam na kontener , który bedzie mial swoj state i rozwiaze nam probmem z tego punktu 2 . Pamietajmy o webpacku.

IV. Funkcjonalność

1. Po stworzeniu funkcji dodającej film mamy ciekawy przypadek zajrzyj do commitu cd947523a593d2b0a9a9cc57448bbd28035d809c

   Odpowiedz na pytanie z komentarza, który znajduje się tutaj:
   movieList.jsx line 13
   To bardzo ważne żeby to rozumieć.

2. Teraz już wiesz , że constructor nie wywołuje się po zmianie props, a to oznacza, że jeśli tam przypiszemy propsy do state, to po ich zmianie z innego komponentu one się już nie zmienią. Potrzebujemy funkcji, która nam je "odświezy"- dlatego użyjemy componentWillReceiveProps.

V. Routing

Routing jest dostępny pod tym commitem:
8625b25aa82a8109f3acce644f3bc918a82498b2

1. Żeby przećwiczyć Routing potrzebujemy:
   a) Ustalić po co nam Routing
   b) Zainstalować odpowiednie paczki dla npm
   c) Ustawienia Routingu

2. ad. a) Ustalmy po co nam Routing.
   Stworzymy sobie dwie dodatkowe podstrony: O Mnie oraz Test - bo nie mam pomysłu na inną.

3. Nie mamy dla nich widoków, ale to nic, będą pustymi stronami z zaślepkami w postaci korpo ipsum.

4. ad. b) Instalujemy odpowiednie paczki dla npm
   npm install react-router@3.0.3 --save

5. Konfigurować webpacka już nie musimy, bo serwuje nam ścieżki z lokalnego serwera lochalhost. Trzeba zaimportować jeszcze w głównym komponencie

rzeczy potrzebne react-routerowi. Zróbmy znowu Refactorig.

6. Wygląda to teraz tak:
   - main renderuje Routing
   - Routing ładuje poszczególne sciezki do podstron
   - app.jsx - to nasza aplikacja do dodawania filmow
   - about.jsx - to podstrona o nas
   - notFound.jsx - to podtrona, która zostaje załadowana jeśli ścieżka nie pasuje do niczego.

7. Dodajmy liinki - czyli stwórzmy komponent odpowiedzialny za nawigacje - navigation.jsx. Tutaj pamiętamy o {this.props.children}

8. Trochę styli do projektu.

VI. FireBase i Google arkusz

a) Firebase - konfiguracja

Przydatne linki :
https://sites.google.com/site/scriptsexamples/new-connectors-to-google-services/firebase/tutorials/read-and-write-data-in-firebase-from-apps-script

https://sites.google.com/site/scriptsexamples/new-connectors-to-google-services/firebase

1. https://console.firebase.google.com
2. Utwórz projekt
3. Z panelu po lewej wybieramy Develop - > Databse - Rozpocznij
4. Tutaj można na testa dodać kilka przykładowych pól, tak ręcznie ale nie trzeba.
5. Tworzymy sobie prywatny dokument Excel, wypelniamy go tak danymi, żeby łatowo je było przetwarzać - czyli nie na Hura, tylko trzeba przemyśleć strukturę.
6. Po wypełnieniu wchodzimy w Narzędzia - Edytor Skryptów
7. Dostajemy pusty dokument, w którym musimy napisać tak skrypt żeby:
   pobrać dane z arkusza ,
   stworzyć odpowiedni obiekt i połączyć się z Firebase i zapisać dane

a) Wklej skrypt, ktory jest nizej.
b) wyciąg secret key : Kliknij po lewej w zebatke -> Ustawienia projektu -> Konta usługi i tam jest -> Klucze tajne bazy danych

c) Nastepnie musze dodac biblioteke firebase do skryptu w tym celu Zasoby -> Biblioteki i tutaj musze podac kod dla tej biblioteki znajdziemy go w drugim linku. Klikamy dodaj, wybieramy najnowsza wersje i potem zapisz.
d) Uruchamiamy zapisujemy i sprawdzamy w Firebase czy sie zapisal obiekt

Zeby wystawic dane na zew, trzeba jeszcze tylko zmienic uprawnienia bazy w tym celu zmien zakladke na Reguly i wpisz :

```
{
  "rules": {
    ".read": true,
    ".write": "auth != null"
  }
}
```

po tym powinien twoj adres byc widoczny z takiego miejsca :
https://movielists-5884a.firebaseio.com/Movies.json

Skrypt

```
---------------------------------------------------------------------------
var firebaseLink = "https://movielists-5884a.firebaseio.com/Movies";
var firebaseSecret = "qoLiLbBq6BsPesGZcyof7z6y9bH4zKfL903mIPhD";

function save_movies() {

  var sheets = SpreadsheetApp.getActiveSpreadsheet().getSheets(); //dostajmy się do arkusza
  var data = []; //do tego obiektu bedziemy zapisywac dane pobrane z arkusza

  for (var i = 0; i < sheets.length; i++) { //lecimy po wszystkich zakładkach/arkuszach czyli w naszym przypadku to Movies
      var sheet = sheets[i]; //podstawiamy kazdy arkusz/zakladke  pod zmienną

      var rows = sheet.getDataRange(); // pobieramy wszystkie wiersze (W dół )
      var numRows = rows.getNumRows(); // pobieramy liczbe wierszy
      var numCols = rows.getNumColumns(); // pobieramy liczbe kolumn

       var values = rows.getValues();   // pobieramy wszytko do tablicy :D

      Logger.log(values) // tak możemy sobie coś wyświetlić, ale wtedy trzeba zajrzec do Wyświetlanie i dzienniki

       for (var j = 2; j < numRows  ; j++) { //Lecimy po wszystkich wierszach ale zaczynam od drugiego
```

```
        var movie = {}; //tworze obiekt
        movie.title = values[j][0]; //zapisuje tytul
        movie.imdb = values[j][1] //zapisuje opinie z imdb

        //Logger.log(movie.title )
          data.push(movie); //zapisujemy obiekty w tablicy
      }

  }

   var firebase = FirebaseApp.getDatabaseByUrl(firebaseLink, firebaseSecret);
   firebase.setData("", data);

}
```

---

b) Firebase - fetch - ściąganie danych

Fetch i sciaganie danych do aplikacji jest dostępne pod tym commitem:
2e848be5c7a1c5e04ff0a88a89ed801d85a7f2e4

1. Przy zaciaganiu danych z bazy - zastanów się do jakiego komponentu chcesz je załadować.
   Do main.jsx? - NIE
   Do app.jsx? - TAK - bo tutaj mamy naszą listę

   Na chwilę obecną ładujemy tutaj. Potem się zastanowimy czy może nie gdzieś indziej.

2. componentDidMount() - pamiętasz prawda?

3. Sprawdź jak wygląda obiekt, który przychodzi