## Схема компонентов приложения React.js с Router и Axios

**Используется**

* React
* react-router-dom
* axios
* bootstrap
* react-bootstrap
* moment

**Структура проекта**

Изображение выглядит как текст

Автоматически созданное описание

Вкратце:

- все лежит в react-crud

- **package.json** автоматически создаваемый файл при старте проекта. Содержит 4 основных модулей: react, react-router-dom, axios и react-bootstrap.

- src/App.js – это главный контейнер, с него начинается запуск нашего приложения и через него имеем выход на другие элементы и страницы. Содержит Router, который читает urls из DRF и преобразует их в любой другой вид в браузере. То есть после реакта вид ссылок может отличаться. Об этом ниже

- src/services содержит весь CRUD: ArticleDataService имеет методы для отправки HTTP-запросов к Apis.

- **http-common.js** инициализирует **axios** с помощью базового URL **-адреса** HTTP и заголовков.

- **.env** настраивает порт для этого приложения React CRUD.

## Настройка проекта React.js

Откройте cmd в папке, в которую вы хотите сохранить папку Project, выполните команду:

npx create-react-app react-crud

Для проверки:

cd react-crud

npm start

Должно появиться тестовое окно в браузере.

После того, как процесс будет завершен, создайте дополнительные папки и файлы в виде следующего дерева, указанного выше.

## Импортировать Bootstrap в приложение React CRUD

Выполнить команду: npm install react-bootstrap bootstrap

Откройте **src** / App.js и измените код внутри него следующим образом:

import React, { Component } from "react";

import "bootstrap/dist/css/bootstrap.min.css";

class App extends Component {

render() {

// ...

}

}

export default App;

## Добавить маршрутизатор React в приложение React CRUD

- Выполните команду: npm install react-router-dom.  
- Откройте **src** / index.js и оберните App компонент по BrowserRouter объекту.

import React from "react";

import ReactDOM from "react-dom";

import { BrowserRouter } from "react-router-dom";

import App from "./App";

import \* as serviceworker from "./ serviceworker ";

ReactDOM.render(

<BrowserRouter>

<App />

</BrowserRouter>,

document.getElementById("root")

);

serviceworker.unregister();

у меня файл serviceworker не создался сам. Скачал его из интернета. Возьмите в проекте.

## Добавить панель навигации в приложение React CRUD

Откройте **src** / App.js , этот Appкомпонент является корневым контейнером для нашего приложения, он будет содержать Switch объект с несколькими Route. Каждый Route является url и одновременно указывает на компонент React. Элемент withRouter нужен для панели меню, он прикрепляет историю ссылок к указанному элементы при адресации и позволяет получить из роутера имя активного элемента, который инициировал ссылку. Тем самым можно выделить активный элемент меню

import 'bootstrap/dist/css/bootstrap.min.css';  
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";  
  
import Header from './Components/Header';  
import Navibar from './Components/Navibar';  
import Footer from './Components/Footer';  
  
  
import Home from './Pages/Home';  
import Start from './Pages/Start';  
import Develop from './Pages/Develop';  
import Admin from './Pages/Admin';  
import Design from './Pages/Design';  
import Management from './Pages/Management';  
import Marketing from './Pages/Marketing';  
import Popsci from './Pages/Popsci';  
import Article from './Pages/Article';  
  
import { withRouter } from "react-router";  
  
const NavibarWithRouter = withRouter(Navibar);  
function App() {  
 return (  
 <>  
 <Router>  
 <Header/>  
 <NavibarWithRouter />  
 <Switch>  
 <Route exact path="/" component={Home} title="Index Page"/>  
 <Route exact path="/start" component={Start} />  
 <Route exact path="/develop" component={Home} />  
 <Route exact path="/admin" component={Home} />  
 <Route exact path="/design" component={Home} />  
 <Route exact path="/management" component={Home} />  
 <Route exact path="/marketing" component={Home} />  
 <Route exact path="/popsci" component={Home} />  
 <Route exact path="/articles/:slug" component={Article} />  
 </Switch>  
 </Router>  
 <Footer />  
 </>  
 );  
}  
  
export default App;

## Инициализировать Axios для HTTP-клиента React CRUD

Давайте установим AXIOS с командой: npm install axios.  
В папке **src** мы создаем файл http-common.js со следующим кодом:

import axios from "axios";

export default axios.create({

baseURL: "http://localhost:8080/api",

headers: {

"Content-type": "application/json"

}

});

Вы можете изменить baseURL это в зависимости от URL-адреса REST API, настроенного вашим сервером. Если DRF запускается на порту <http://localhost:8080> и в главном urls.py поекта стоит path('api/', include('main\_app.urls')), то ловить данные из DRF будем на:

baseURL: <http://localhost:8080/api>

**Настроить CORS**

Это служба ставится в DRF. Она обратная axios. Нам нужно разрешить запросы к нашему приложению Django из других источников.  
В этом примере мы собираемся настроить CORS для приема запросов от localhost:8081, то есть react запускается на порту 8081.

Сначала установите библиотеку *django-cors-headers* :  
pip install django-cors-headers

В *settings.py* добавьте конфигурацию для CORS:

INSTALLED\_APPS = [

...

# CORS

'corsheaders',

]

Вам также необходимо добавить класс промежуточного программного обеспечения для прослушивания ответов:

MIDDLEWARE = [

...

# CORS

'corsheaders.middleware.CorsMiddleware',

'django.middleware.common.CommonMiddleware',

]

***Примечание:*** CorsMiddleware следует размещать как можно выше, особенно перед промежуточным программным обеспечением, которое может генерировать ответы, такие какCommonMiddleware.

Затем установите *CORS\_ORIGIN\_ALLOW\_ALL* и добавьте хост в *CORS\_ORIGIN\_WHITELIST* :

CORS\_ORIGIN\_ALLOW\_ALL = False

CORS\_ORIGIN\_WHITELIST = (

'http://localhost:8081',

)

* *CORS\_ORIGIN\_ALLOW\_ALL* : Если True все источники будут приняты (не используйте белый список ниже). По умолчанию False.
* *CORS\_ORIGIN\_WHITELIST* : список источников, которым разрешено выполнять межсайтовые HTTP-запросы. По умолчанию [].

## Создать службу данных

На этом этапе мы собираемся создать службу, которая использует объект axios, указанный выше, для отправки HTTP-запросов.

**сервисы** / article.service.js

import http from "../http-common";  
  
class ArticleDataService {  
 getAll() {  
 return http.get("/articles/");  
 }  
  
 get(slug) {  
 return http.get(`/articles/${slug}`);  
 }  
  
 create(data) {  
 return http.post("/articles", data);  
 }  
  
 update(id, data) {  
 return http.put(`/articles/${id}`, data);  
 }  
  
 delete(id) {  
 return http.delete(`/articles/${id}`);  
 }  
  
 deleteAll() {  
 return http.delete(`/articles`);  
 }  
  
 findByTitle(title) {  
 return http.get(`/articles?title=${title}`);  
 }  
}  
  
export default new ArticleDataService();

Мы называем axios get, post, put, delete метод , соответствующий HTTP запросы: GET, POST, PUT, DELETE , чтобы сделать CRUD операции. Ссылка "../http-common"; указывает на файл адресации axios, описанный выше.

## Компоненты

src/Pages – страницы проекта

src/Components – отдельные элементы страниц (шаблоны)

При запуске главной страницы в App.js получаем ссылку на главную страницу и по ней берет данные из DRF. Далее роутер рендерит соответствующую страницу Home, которая указана в роутера.

Страница Home получает из роутера данные со списком статей и перебирая каждый элемент файла данных подставляет их в шаблон Article.

Остальное по аналогии. Смотрим App.js, в нем роутер и видим какая страница шаблона сопоставлена каждому из url. Идем в шаблон, там весь программный код.

## Дополнительно

Ставим styled-components.com, он позволяет изменять стандартные css бутстрапа:

npm install --save styled-components

Чтобы запустить APP на другом порту (например 3002), выполните следующую команду создать файл .env с содержимым PORT=3002

И ещё для красивого вывода даты поставим такое:

npm i --save-dev moment

Применение:

const data = moment(this.props.article.date\_update).format('MMMM Do YYYY')

Для изучения рекомендую:

<https://www.youtube.com/watch?app=desktop&v=jT8wO8aKea0&list=PLVfMKQXDAhGWamOcqqIsMAbsP6pl0vUtW&index=1>

<https://www.youtube.com/watch?v=VyeZSerAoh4&t=6s>

Коротко и емко про верстку Reacta.

А тут как собираются вместе DRF и React:

DRF: <https://bezkoder.com/django-react-axios-rest-framework/#Configure_CORS>

React: <https://bezkoder.com/react-crud-web-api/>