В этом уроке мы с вами попробуем использовать для получения данных реальный API. Для этого нам нужно его создать.

Так как реального API у меня нет, то я нашел в гугле простой сервис, чтобы писать тестовые запросы. Он называется mocky.io. Вот так выглядит страничка мы можем просто добавить в body наши телефоны в виде массива. Единственный нюанс в том, чтобы нажать клавишу advanced mode и указать хедер Access-Control-Allow-Origin: \* Без него наши запросы работать не будут.

Я приготовил те данные телефонов, которые мы использовали в виде JSON.

{

"phones": [

{

"id": "1",

"categoryId": "1",

"name": "Apple iPhone 5c",

"description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ultricies lorem odio, at laoreet tellus sodales in. Nullam maximus eros ut tortor ultricies rutrum. Aliquam euismod lacus non est egesta",

"price": 823,

"image": "/uploads/iphone5c-selection-hero-2013.png",

"cpu": "1.3GHz Apple A6",

"camera": "8mp (3264x2448)",

"size": "124.4mm x 59.2mm x 8.97mm (4.9 x 2.33 x 0.35)",

"weight": "132 grams (4.7 ounces) with battery",

"display": "4.0 326 pixel density",

"battery": "1480 mAh",

"memory": "16GB, 32GB and RAM 1 GB"

},

{

"id": "2",

"categoryId": "1",

"name": "Apple iPhone 6",

"description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ultricies lorem odio, at laoreet tellus sodales in. Nullam maximus eros ut tortor ultricies rutrum. Aliquam euismod lacus non est egesta",

"price": 953,

"image": "/uploads/51u6y9Rm8QL.\_SY300\_.jpg",

"cpu": "1.3GHz Apple A6",

"camera": "8mp (3264x2448)",

"size": "124.4mm x 59.2mm x 8.97mm (4.9 x 2.33 x 0.35)",

"weight": "132 grams (4.7 ounces) with battery",

"display": "4.0 326 pixel density",

"battery": "1480 mAh",

"memory": "16GB, 32GB and RAM 1 GB"

},

{

"id": "3",

"categoryId": "4",

"name": "Lenovo A6000",

"description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ultricies lorem odio, at laoreet tellus sodales in. Nullam maximus eros ut tortor ultricies rutrum. Aliquam euismod lacus non est egesta",

"price": 764,

"image": "/uploads/\_35%20(1).JPG",

"cpu": "1.3GHz Apple A6",

"camera": "8mp (3264x2448)",

"size": "124.4mm x 59.2mm x 8.97mm (4.9 x 2.33 x 0.35)",

"weight": "132 grams (4.7 ounces) with battery",

"display": "4.0 326 pixel density",

"battery": "1480 mAh",

"memory": "16GB, 32GB and RAM 1 GB"

},

{

"id": "4",

"categoryId": "5",

"name": "Nokia Lumia 1520",

"description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ultricies lorem odio, at laoreet tellus sodales in. Nullam maximus eros ut tortor ultricies rutrum. Aliquam euismod lacus non est egesta",

"price": 674,

"image": "/uploads/Lumia1520-Front-Back-png.png",

"cpu": "1.3GHz Apple A6",

"camera": "8mp (3264x2448)",

"size": "124.4mm x 59.2mm x 8.97mm (4.9 x 2.33 x 0.35)",

"weight": "132 grams (4.7 ounces) with battery",

"display": "4.0 326 pixel density",

"battery": "1480 mAh",

"memory": "16GB, 32GB and RAM 1 GB"

},

{

"id": "5",

"categoryId": "3",

"name": "HTC One",

"description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ultricies lorem odio, at laoreet tellus sodales in. Nullam maximus eros ut tortor ultricies rutrum. Aliquam euismod lacus non est egesta",

"price": 674,

"image": "/uploads/htc-one-m7-802w-dual-sim-silver.jpg",

"cpu": "1.3GHz Apple A6",

"camera": "8mp (3264x2448)",

"size": "124.4mm x 59.2mm x 8.97mm (4.9 x 2.33 x 0.35)",

"weight": "132 grams (4.7 ounces) with battery",

"display": "4.0 326 pixel density",

"battery": "1480 mAh",

"memory": "16GB, 32GB and RAM 1 GB"

},

{

"id": "6",

"categoryId": "2",

"name": "Samsung Galaxy S6",

"description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ultricies lorem odio, at laoreet tellus sodales in. Nullam maximus eros ut tortor ultricies rutrum. Aliquam euismod lacus non est egesta",

"price": 674,

"image": "/uploads/Agnes\_Case\_for\_Samsung\_Galaxy\_S6\_(1)\_\_92643\_thumb.jpg",

"cpu": "1.3GHz Apple A6",

"camera": "8mp (3264x2448)",

"size": "124.4mm x 59.2mm x 8.97mm (4.9 x 2.33 x 0.35)",

"weight": "132 grams (4.7 ounces) with battery",

"display": "4.0 326 pixel density",

"battery": "1480 mAh",

"memory": "16GB, 32GB and RAM 1 GB"

}

]

}

Добавляем body и header и нажимаем сгенерировать. Нам выдало ссылку, по которой доступны наши данные. Теперь мы можем использовать это API в нашем приложении.

Так как мы сделали хорошую архитектуру для екшенов и API изменения в коде будут минимальны. Для работы с апи я предпочитаю использовать библиотеку superagent. Но вы можете использовать любую на ваш выбор, например axios или fetch, код от этого изменится минимально.

Давайте в файл api/index.js импортируем superagent

import request from 'superagent'

Теперь мы можем в fetchPhones написать

export const fetchPhones = async () => {

const {body} = await request.get(

'http://www.mocky.io/v2/5918b9461200001f1040dbeb'

)

return body.phones

}

То есть мы просто говорим superagent сделай нам get и с помощью await мы ждем результата. В результате мы получаем обьект у которого есть поле body, который содержит наш json. Так как у нас все поля лежат в поле phones, то мы просто пишем return body.phones.

Если мы посмотрим в браузер, то у нас все продолжает работать, но в network мы теперь видим запрос к нашему API. Весь остальной код нам менять не нужно, так как он работает абсолютно также.

Если у вас что-то не получается, либо возникли вопросы и комментарии, пишите их прямо под этим видео и я обязательно на них отвечу.