OneBitFood V2 – Aula 2

{} onebitcode.com/onebitfood-v2-aula-2

[Aviso importante: Caso você copie e cole os códigos, alguns caracteres invisíveis podem vir junto e dar alguns Bugs, então remova os espaços entre as linhas (e os crie novamente com a barra)]

Bons códigos

0 – Links importantes

- Guia para instalar as dependências: https://onebitcode.com/guia-de-instalacao-do-ruby-on-rails
- API completa no GitHub: https://github.com/OneBitCodeBlog/onebitfoodV2-api
- Cliente Web no GitHub: https://github.com/OneBitCodeBlog/onebitfoodV2-nextjs

1 – Planejamento

A ideia inicial

Criar um APP inspirado no Ifood usando Ruby On Rails como API e Next.js como cliente Web

O Planejamento

1. Mockups
    1. Ferramenta: https://www.figma.com
    2. Material: https://drive.google.com/drive/folders/1mqVAQP-96Xa4I8AcdsuFc0v_XJ_elqVx?usp=sharing
2. Modelo do banco de dados
    1. Ferramenta: DbDesigner
    2. Material: https://dbdesigner.page.link/RvxXdi581QjxiDqe8
3. Documentação dos endpoints
    1. Ferramenta: https://www.postman.com
    2. Material: https://documenter.getpostman.com/view/10378249/TzRPk9yD

Dependências

A seguir, veja as dependências para este projeto (cliente Web)

- Next.js 10.1.3
- React 17.0.2
- React-bootstrap 1.5.2
- Bootstrap 5.0.0
- Swr 0.5.5
- Sass 1.32.8
- Recoil ^0.2.0
- Recoil Persist ^2.5.0
- Slick Carousel ^1.8.1

2 – Criação do projeto

Nesta aula nós vamos gerar o nosso projeto básico usando o Create React APP.

1 – Crie um novo projeto ReactJS. Para isso abra seu terminal e digite:

Default

```
1  npx create-next-app onebitfood-client
```

2 – Após isso, acesse o diretório do projeto.

Default

```
1  cd onebitfood-client
```

3 – Para testar que seu projeto funcionou rode:

Default

```
1  npm run dev
```

4 – Apague o arquivo styles/Home.module.css

5 – Atualize o arquivo _app.js:

Default

```
import Head from 'next/head';
import '../styles/globals.scss';

export default function MyApp({ Component, pageProps }) {
 return(
   <>
     <Head>
       <title>OneBitFood V2</title>
     </Head>

     <main>
       <Component {...pageProps} />
     </main>
   </>
)
}
```

5 – Limpe o arquivo **pages/index.js** :

Default

```
export default function Home() {
 return (
   <>
   </>
)
}
```

> Assim teremos um código menos verboso nesse inicio removendo arquivos que não serão utilizados.

6 – Atualize no arquivo package.json o script dev:

Default

```
1  "dev": "next dev -p 3001",
```

3 – Instalação de dependências

Nesta aula nós vamos instalar as principais dependências do projeto como Swr para as chamadas web, sass para utilizadas o sass ao invés do css comum e o Bootstrap para a estilização do projeto.

1 – Instale também o Swr para fazermos requisições em Apis:

Default

```
1  yarn add swr@0.5.5
```

2 – Instale o node-sass, para que possamos adicionar SCSS no nosso projeto

Default

```
1  yarn add sass@1.32.12
```

3 – Instale o framework CSS Bootstrap

Default

```
1  yarn add react-bootstrap@1.5.2 bootstrap@5.0.0
```

4 – Instale a biblioteca de ícones:

Default

```
1  yarn add react-icons@4.2.0
```

4 – Estrutura do Projeto e estilos globais

Para manter a organização vamos criar a estrutura principal de diretórios do nosso projeto já no inicio e também vamos criar alguns estilos comuns a todos os elementos do SPA como cores, espaçamentos e imagem de background.

1 – Crie a de components do nosso projeto:

Default

```
mkdir components
```

2 – Baixe as seguintes imagens e as coloque dentro da pasta `public` : https://drive.google.com/drive/folders/1-iHaCLfuyYktwen46bOMnxSF3dRD22XD?usp=sharing

3 – Limpe o conteúdo do arquivo styles/globals.css e renomeie ele para globals.scss

4 – Atualize o arquivo pages/_app.js colocando:

Default

```
import Head from 'next/head';
import '../styles/globals.scss';

export default function MyApp({ Component, pageProps }) {
 return(
   <>
     <Head>
       <title>OneBitFood V2</title>
       <link rel="icon" href="/favicon.ico" />
     </Head>

     <main>
       <Component {...pageProps} />
     </main>
   </>
)
}
```

5 – Crie um arquivo chamado styles/colors.scss coloque nele:

Default

```
$custom-red: #FF0043;
$custom-gray: #cccccc;
$custom-gray-darker: #666666;
$custom-black: #353535;
$custom-orange: #F7C97C;
```

6 – Em globals.scss coloque:

Default

```
@import "colors.scss";

$theme-colors: (
 "custom-red": $custom-red,
 "custom-gray": $custom-gray,
 "custom-gray-darker": $custom-gray-darker,
 "custom-black": $custom-black,
 "custom-orange": $custom-orange
);
```

7 – Também adicione uma extensão da variável spacer do bootstrap para termos mais tamanhos de margins e padings e também o efeito de bold para font-weight 600 disponíveis:

Default

```
$spacer: 1rem;
$spacers: (
 0: 0,
 1: ($spacer * .25),
 2: ($spacer * .5),
 3: $spacer,
 4: ($spacer * 1.5),
 5: ($spacer * 3),
 6: ($spacer * 5),
 7: ($spacer * 10),
 8: ($spacer * 15),
 9: ($spacer * 20)
);

$font-weight-bold: 600;
```

8 – Agora vamos importar o Bootstrap no mesmo arquivo (globals.scss):

Default

```
@import "~bootstrap/scss/bootstrap";
```

Obs: Coloque esse import depois do $spacers.

9 – Para incluir o Background acrescente em globals.scss:

Default

```
html, body {
 margin: 0px;
 padding: 0px;
 height: 100%;
}

body {
 background-image: url('../public/bg.png');
}
```

> Aqui nós incluímos o background fixo da nossa aplicação.

10 – Para mudar definir a fonte padrão inclua a importação da fonte e atualize os estilos do “html, body” incluídos anteriormente em globals.scss:

Default

```
@import url('https://fonts.googleapis.com/css2?family=Lexend:wght@300;400;500;600;700&display=swap');


html, body {
 margin: 0px;
 padding: 0px;
 height: 100%;
 font-family: 'Lexend', sans-serif;
}
```

11 – Para alterar o comportamento da classe container do bootstrap para que o conteúdo ocupe mais espaço lateral na tela no mobile, coloque em globals.scss:

Default

```
@media (max-width:1025px) { .container { width: 95% !important;} }
```

12 – No arquivo pages/index.js, coloque:

Default

```
import Container from 'react-bootstrap/Container';

export default function Home() {
  return (
    <>
      <Container>
        Home Page Content
      </Container>
    </>
  )
}
```

13 – Suba o projeto e vejo o background + bootstrap em ação

Default

```
yarn dev
```

5 – Criando o component Header

Nesta parte, vamos criar o componente **Header** que será utilizado em toda aplicação que será o **Header**.

1 – Primeiro, vamos criar o diretório `src/components/header` e dentro dele o arquivo `index.js`

Default

```
mkdir components/Header
touch components/Header/index.js
```

2 – No arquivo components/Header/index.js` coloque:

Default

```
import React from 'react';

import { Navbar } from 'react-bootstrap';
import Image from 'next/image';
import Link from 'next/link';

const Header = () => {
 return (
   <Navbar bg="white" expand="lg" className="border-bottom border-custom-gray">
     <Navbar.Brand>
       <Link href="/restaurants">
     <a>
         <Image
           src="/logo.png"
           alt="OneBitFood"
           width={200}
           height={44}
         />
       </a>
       </Link>
     </Navbar.Brand>
   </Navbar>
)
}

export default Header;
```

3 – Agora vamos adicionar o Header (e o Container) no nosso app. Importe e adicione o componete Header em `src/_app.js`

Default

```
import Head from 'next/head';
import '../styles/globals.scss';
import Header from "../components/Header";
import Container  from 'react-bootstrap/Container';

export default function MyApp({ Component, pageProps }) {
 return(
   <>
     <Head>
       <title>OneBitFood V2</title>
       <link rel="icon" href="/favicon.ico" />
     </Head>

     <main>
       <Header />
       <Container className="mt-5">
         <Component {...pageProps} />
       </Container>
     </main>
   </>
)
}
```

4 – Verifique com ficou subindo o projeto:

Default

```
yarn dev
```

5 – Para incluir uma classe geral que vai dar destaque ao que for clicável:

a – No globals.scss coloque:

Default

```
.clickable_effect:hover {
 opacity: 0.8;
 transform:scale(1.01);
 cursor: pointer;
}
```

b – Na imagem do Header inclua a classe:

Default

```
<Image
 src="/logo.png"
 alt="OneBitFood"
 width={200}
 height={44}
 className="clickable_effect"
/>
```

6 – Suba o projeto, passe o mouse sobre o logo e veja o efeito.

6 – Criando a Home

1 – Adicione em `pages/index.js`

Default

```
import { Button, Row, Col } from 'react-bootstrap';
import Link from 'next/link';
import { FaCrosshairs } from 'react-icons/fa';

export default function Home() {
 return (
   <Row className="mt-8 justify-content-center">
     <Col md={7} xs={12} className="text-center">
       <h1 className='fw-bolder text-custom-gray-darker mb-5 lh-base display-5'>
        Comida saudável e gostosa direto na sua casa
       </h1>
       <Link href='/restaurants'>
         <Button variant="custom-red" size="lg" className='text-white'>
           <FaCrosshairs className='pb-1'/>
           <span className='px-2 fw-bolder'>ENCONTRAR AGORA</span>
         </Button>
       </Link>
     </Col>
   </Row>
  )
}
```

7 – Efeito de digitação

Vamos incluir na nossa Home um efeito de “digitando” para tornar a experiencia do usuário ainda melhor

1 – Crie um component em components/Typewriter/index.js

2 – Crie a estrutura do Component:

Default

```
1  import React from 'react';
2
3  export default function Typewriter(props) {
4
5   return(
6     <></>
7  )
8  }
```

3 – Crie um estado para armazenar o que vai aparecer na página:

Default

```
1  import React, { useState } from 'react';
2  ...
3
4  export default function Typewriter(props) {
5   const [phrase, setPhrase] = useState("")
6   ...
7     <>{phrase}</>
8   ...
```

4 – Inclua um “efeito”:

Default

```
import React, { useState, useEffect } from 'react';
...

export default function Typewriter(props) {
 ...

 useEffect(() => {
   let currentText = "";
   props.text.split('').forEach((char, index) => {
     setTimeout( () => {
       currentText = currentText.slice(0, -1)
       currentText += char;
       if(props.text.length != index + 1) currentText += "▮"
       setPhrase(currentText)
     }, 200 + (index * 100));
   })
}, []);
```

5 – Atualize o component pages/index.js para colocar o Typewriter:

Default

```
import { FaCrosshairs } from 'react-icons/fa';
import { Container, Button, Row, Col } from 'react-bootstrap';
import Link from 'next/link';
import Typewriter from '../components/Typewriter';

export default function Home() {
 return (
   <Row className="mt-8 justify-content-md-center">
     <Col md={7} xs={12} className='text-center'>
       <h1 className='fw-bolder text-custom-gray-darker mb-5 lh-base display-5'>
         <Typewriter text="Comida saudável e gostosa direto na sua casa"/>
       </h1>
       <Link href="/restaurants">
         <Button variant="custom-red" size="lg" className='text-white'>
           <FaCrosshairs/>
           <span className='px-2'>ENCONTRAR AGORA</span>
         </Button>
       </Link>
     </Col>
   </Row>
  )
}
```

8 – Listando Restaurantes

1- Crie um diretório `src/screens/restaurants` e dentro dele o arquivo `index.js`

Default

```
mkdir pages/restaurants
touch pages/restaurants/index.js
```

2 – Crie a estrutura da página:

Default

```
export default function Restaurants() {
 return (
   <>

   </>
)
}
```

3 – Agora vamos criar um componente para a lista de restaurantes desta tela:

Default

```
mkdir components/ListRestaurants
touch components/ListRestaurants/index.js
```

4 – Adicione o seguinte conteúdo no component criado:

Default

```
import React from 'react';
import { Container, Row, Col }  from 'react-bootstrap';

export default function ListRestaurants() {

 return (
  <div className='mt-5'>
    <h3 className='fw-bold'>Restaurantes</h3>
    <Row>
      <Col>Restaurants...</Col>
    </Row>
  </div>
 )
}
```

5 – Volte no arquivo `pages/restaurants/index.js` e altere o conteúdo para:

Default

```
import ListRestaurants from "../../components/ListRestaurants";

export default function Restaurants() {
 return (
  <>
    <ListRestaurants />
  </>
)
}
```

6 – Agora vamos criar dois métodos úteis que serão reutilizados depois, o toCurrency para formatar o valores financeiros e o trucanteString para exibir no máximo X caracteres de uma string.

a – Crie uma pasta chamada services, rodando:

Default

```
mkdir services
```

b – Dentro dela crie o arquivo toCurrency.js e coloque nele:

Default

```
export default function toCurrency(value) {
 const formatter = new Intl.NumberFormat('pt-BR', {
   style: 'currency',
   currency: 'BRL',
})
 return formatter.format(value)
}
```

c – Crie dentro de services também o arquivo trucanteString.js e coloque nele:

Default

```
export default function truncateString(str, num) {
 if (str.length <= num)
   return str
else
   return str.slice(0, num) + '...'
}
```

7 – Vamos criar o component restaurant que será o box usado para mostrar as informações de cada restaurant no listagem

Default

```
mkdir components/ListRestaurants/Restaurants
touch components/ListRestaurants/Restaurants/index.js
```

8 – Adicione o seguinte conteúdo no component criado:

Default

```
1   import React from 'react';
2   import { Row, Col, Card } from 'react-bootstrap';
3   import { FaStar } from 'react-icons/fa';
4   import Image from 'next/image'
5   import Link from 'next/link'
6   import toCurrency from '../../../services/toCurrency';
7   import truncateString from '../../../services/truncateString';
8
9   const Restaurant = (props) => (
10    <Col lg={6} sm={6} xs={12} className="mb-4">
11      <Link href={`restaurants/${props.id}`}>
12        <Card body className='clickable_effect'>
13          <Row>
14            <Col md={5} xs={12}>
15              <Image
16                src={props.image_url}
17                alt={props.name}
18                width={300}
19                height={200}
20                layout="responsive"
21              />
22            </Col>
23            <Col md={5} xs={10}>
24              <h5>{truncateString(props.name, 25)}</h5>
25              <p className='mb-1'>
26                <small> {truncateString(props.description, 60)} </small>
27              </p>
```

```
28              <p>
29                <small className='fw-bold'>{props.category_title}</small>
30              </p>
31              <small className='border px-3 border-custom-gray fw-bold'>
32                entrega {toCurrency(props.delivery_tax)}
33              </small>
34            </Col>
35            <Col md={2} xs={2} className="text-center">
36              <span className='text-custom-orange'>
37                <FaStar/> 5
38              </span>
39            </Col>
40          </Row>
41        </Card>
42      </Link>
43    </Col>
44  )
45
46  export default Restaurant;
```

9 – Atualize o component src/components/ListRestaurants/index.js para renderizarmos uma lista de restaurants:

Default

```
1   import React from 'react';
2   import { Container, Row, Col }  from 'react-bootstrap';
3   import Restaurant from '../ListRestaurants/Restaurant';
4
5   export default function ListRestaurants() {
6    const restaurants = [
7     {
```

```
8       'id': 1,
9       'name': 'example 1',
10      'description': 'Javascript Ipsum, Javascript Ipsum e Javascript Ipsum',
11      'delivery_tax': '5',
12      'image_url': '/restaurant.jpeg',
13      'category_title': 'Cozinha japonesa'
14    },
15    {
16      'id': 2,
17      'name': 'example 2',
18      'delivery_tax': '10',
19      'description': 'Javascript Ipsum, Javascript Ipsum e Javascript Ipsum',
20      'image_url': '/restaurant.jpeg',
21      'category_title': 'Cozinha mineira'
22    },
23    {
24      'id': 3,
25      'name': 'example 3',
26      'delivery_tax': '15',
27      'description': 'Javascript Ipsum, Javascript Ipsum e Javascript Ipsum',
28      'image_url': '/restaurant.jpeg',
29      'category_title': 'Cozinha vegana'
30    },
31    {
32      'id': 4,
33      'name': 'example 4',
34      'delivery_tax': '10',
35      'description': 'Javascript Ipsum, Javascript Ipsum e Javascript Ipsum Javascript
    Ipsum, Javascript Ipsum e Javascript Ipsum',
36
        'image_url': '/restaurant.jpeg',
37
```

```
38        'category_title': 'Cozinha vegana'
39      }
40    ]
41
42     return (
43       <div className='mt-5'>
44         <h3 className='fw-bold'>Restaurantes</h3>
45         <Row>
46          {restaurants.map((restaurant, i) => <Restaurant {...restaurant} key={i}/>)}
47         </Row>
48       </div>
49     )
       }
```

10 – Veja o que foi feito subindo o projeto e acessando /restaurants:

Default

```
1  yarn dev
```

9 – Baixando os restaurantes da API

1 – Uma pequena correção, na API, na partial _restaurante acrescente (faltou devolvermos esse campo importante):

Default

```
1  json.category_title restaurant.category.title
```

2 – Crie um arquivo na raiz do projeto chamado next.config.js e coloque nele:

Default

```
module.exports = {
 images: {
   domains: ['localhost'],
},
 env: {
   apiUrl: 'http://localhost:3000',
},
}
```

ps: Reinicie o servidor se ele estiver ativo

3 – Para fazer as chamadas do restaurante crie um service chamado getRestaurants.js e coloque nele:

Default

```
import useSWR from 'swr';

export default function getRestaurants() {
 const fetcher = (...args) => fetch(...args).then((res) => res.json());

 const { data, error } = useSWR(
   `${process.env.apiUrl}/api/restaurants`,
   fetcher,
  { revalidateOnFocus: false }
)

 return { restaurants: data, isLoading: !error && !data, isError: error }
}
```

4 – No component ListRestaurants inclua essa chamada atualizando o conteúdo dele para:

Default

```
1  import React from 'react';
2  import { Row, Col, Spinner, Alert } from 'react-bootstrap';
3  import Restaurant from '../ListRestaurants/Restaurant';
4  import getRestaurants from '../../serivces/getRestaurants';
5
6  export default function ListRestaurants() {
7   const { restaurants, isLoading, isError } = getRestaurants();
8
9    function renderContent() {
10     if(isError)
11       return <Col><Alert variant='custom-red'>Erro ao carregar</Alert></Col>
12     else if(isLoading)
13       return <Col><Spinner animation='border'/></Col>
14     else if(restaurants.lenght == 0)
15       return <Col>Nenhum restaurante disponível ainda...</Col>
16     else
17       return restaurants.map((restaurant, i) => <Restaurant {...restaurant} key={i}/>)
18 }
19
20   return (
21     <div className='mt-5'>
22       <h3 className='fw-bold'>Restaurantes</h3>
23       <Row>
24        {renderContent()}
25       </Row>
26     </div>
27    )
28 }
```

10 – Desafio dos Reviews

1 – Crie uma tabela na API para armazenar os reviews que o restaurante recebeu.

2 – Associe essa tabela ao resturante.

3 – Popule ela com reviews fake nos seeds.

4 – Devolva a nota média do restaurante junto com os outros dados dele.

5 – Exiba ela no frontend (onde colocamos o valor default dos reviews para 5).

11 – Lista de Categorias

Nesta aula nós vamos desenvolver o component que mostrada a lista de categorias para filtrarmos os restaurantes.

1 – Crie uma pasta `src/components/categories` e o arquivo `index.js` para o componente `Categories`

Default

```
mkdir components/Categories
touch components/Categories/index.js
```

2 – Para este componentes, primeiro é necessário instalar o plugin Slick Carousel e seu sua adaptação para o React

Default

```
yarn add slick-carousel react-slick
```

3 – Dentro da pasta `src/components/categories` crie um arquivo `slick_settings.js` para as configurações do Slick com o seguinte conteúdo

Default

```
export default {
 speed: 500,
 slidesToShow: 5,
 slidesToScroll: 4,
 initialSlide: 0,
 dots: false,
```

```
7     infinite: false,
8     responsive: [
9      {
10       breakpoint: 1024,
11       settings: {
12         slidesToShow: 3,
13         slidesToScroll: 3,
14       }
15     },
16     {
17       breakpoint: 600,
18       settings: {
19         slidesToShow: 2,
20         slidesToScroll: 2,
21         initialSlide: 2
22       }
23     },
24     {
25       breakpoint: 480,
26       settings: {
27         slidesToShow: 2,
28         slidesToScroll: 1,
29         dots: true
30       }
31     }
32   ]
33   };
```

4 – No component categories, crie a base colocando:

Default

```
1   import Slider from "react-slick";
2   import "slick-carousel/slick/slick.css";
3   import "slick-carousel/slick/slick-theme.css";
4   import { Card, Container, Row, Col }  from 'react-bootstrap';
5   import slickSettings from "./slick_settings";
6
7   export default function Categories() {
8
9    return (
10     <>
11       <h3 className='fw-bold'>Categorias</h3>
12       <Card className="mt-2">
13         <Slider {...slickSettings} className="px-4 pt-4 text-center">
14         </Slider>
15       </Card>
16     </>
17    )
18  }
```

5 – Atualize a screen Restaurants `pages/restaurants/index.js`

Default

```
1   import ListRestaurants from "../../components/ListRestaurants";
2   import Categories from "../../components/Categories";
3
4   export default function Restaurants() {
5    return (
6      <>
7        <Categories/>
8        <ListRestaurants />
9      </>
10    )
11  }
```

7 – Crie o component Category (components/categories/Category/index.js) e coloque nele:

Default

```
import Link from 'next/link';
import Image from 'next/image';

export default function Category(props) {
 return(
   <div>
     <Link href={`/restaurants?category=${props.title}`}>
       <a>
         <Image
           src={props.image_url}
           alt={props.title}
           width={300}
           height={200}
           className='px-1 clickable_effect'
         />
       </a>
     </Link>
     <p className='fw-bold'>{props.title}</p>
   </div>
  )
}
```

8 – Atualize o components Categories importando o Category e passando dados Fake:

Default

```
import Slider from "react-slick";
import "slick-carousel/slick/slick.css";
import "slick-carousel/slick/slick-theme.css";
import { Card, Container, Row, Col }  from 'react-bootstrap';
import slickSettings from "./slick_settings";
```

```
6   import Category from './Category';
7
8   export default function Categories() {
9    const categories = [{
10       'id': 1,
11       'title': 'Italiana',
12       'image_url': '/category.jpeg'
13    },{
14       'id': 1,
15       'title': 'Italiana',
16       'image_url': '/category.jpeg'
17    },{
18       'id': 1,
19       'title': 'Italiana',
20       'image_url': '/category.jpeg'
21    },{
22       'id': 1,
23       'title': 'Italiana',
24       'image_url': '/category.jpeg'
25    },{
26       'id': 1,
27       'title': 'Italiana',
28       'image_url': '/category.jpeg'
29    }
30  ]
31
32   return (
33     <>
34       <h3 className='fw-bold'>Categorias</h3>
35       <Card className="mt-2">
```

```
36          <Slider {...slickSettings} className="px-4 pt-4 text-center">
37           {categories.map((category, i) => <Category {...category} key={i}/>)}
38          </Slider>
39        </Card>
40      </>
41    )
42  }
```

9 – Veja como ficou:

Default

```
1  yarn dev
```

12 – Baixando as categorias da API

Assim como fizemos com restaurantes na aula passada, agora vamos conectar a API para carregar as categorias

1 – Crie um service chamado getCategories.js e coloque nele:

Default

```
1   import useSWR from 'swr';
2
3   export default function getCategories() {
4    const fetcher = (...args) => fetch(...args).then((res) => res.json())
5
6    const { data, error } = useSWR(`${process.env.apiUrl}/api/categories`,
7      fetcher, { revalidateOnFocus: false }
8   )
9
10   return { categories: data, isLoading: !error && !data, isError: error }
11  }
```

2 – No component Categories:

a – Importe o getCategories:

Default

```
1  import getCategories from '../../serivces/getCategories';
```

b – Remova a const categories.

c – Inclua a chamada ao getCategories no component:

Default

```
1  const { categories, isLoading, isError } = getCategories();
```

d – Crie um método para renderizar o erro, a mensagem de carregando e o conteúdo:

Default

```
function renderContent() {
  if(isError)
    return <Alert variant='custom-red'>Erro ao carregar</Alert>;
  else if(isLoading)
    return <Spinner animation="border" />;
  else {
    if(categories.length == 0)
      return <p>Nenhuma categoria disponível ainda</p>;
    else {
      return (
        <Card className="mt-2">
          <Slider {...slickSettings} className="px-4 pt-4 text-center">
           {categories.map((category, i) => <Category category={category} key={i}/>)}
          </Slider>
        </Card>
      )
    }
  }
}
```

e – Atualize as importações dos elementos do bootstrap:

Default

```
import { Card, Container, Spinner, Alert }  from 'react-bootstrap';
```

f – Atualize o conteúdo do método return:

Default

```
return (
 <>
   <h3 className='fw-bold'>Categorias</h3>
 {renderContent()}
 </>
)
```

3 – Atualize o getRestaurants.js para permitir o filtro por categoria:

Default

```
import useSWR from 'swr';
import { useRouter } from 'next/router';
const fetcher = (...args) => fetch(...args).then((res) => res.json())

export default function <span class="md-plain">getRestaurants</span>() {
 const router = useRouter();
 const { category } = router.query;

 let params = '';
 if(category)
  params = `${params == '' ? '?' : '&'}category=${category}`

 const { data, error } = useSWR(
  `${process.env.apiUrl}/api/restaurants${params}`,
  fetcher, {
   revalidateOnFocus: false
  }
 )

 return { restaurants: data, isLoading: !error && !data, isError: error }
}
```

4 – Teste o APP:

Default

```
yarn dev
```

13 – Criando a Busca

Nesta aula vamos criar o componente de busca que vai ficar no cabeçalho do app

1- Crie o diretório `components/SearchBox` e o arquivo `index.js`

Default

```
1 mkdir components/SearchBox
2 touch components/SearchBox/index.js
```

2- Acrescente o seguinte conteúdo no component criado:

Default

```
1  import React, { useState } from 'react';
2  import { Form, Button } from 'react-bootstrap';
3  import { FaSearch } from 'react-icons/fa';
4  import { useRouter } from 'next/router';
5
6  export default function SearchBox() {
7   const [query, setQuery] = useState("")
8   const router = useRouter()
9
10  async function Search(event) {
11    event.preventDefault();
12    router.push(`/restaurants?q=${query}`)
13 }
14
15  return (
16    <Form className='d-flex mx-5 my-2' onSubmit={(e) => Search(e)}>
17      <Form.Control
18        type="text"
19        placeholder="Buscar Restaurantes..."
20        value={query}
21        onChange={(e) => setQuery(e.target.value)}
22        className="me-2"
23      />
24      <Button variant="outline-custom-red" type="submit">
25        <FaSearch />
26      </Button>
27    </Form>
28   )
29 }
```

3 – Atualize o código do `components/Header/index.js` para

Default

```
import React from 'react';

import { Navbar, Nav } from 'react-bootstrap';
import Image from 'next/image';
import Link from 'next/link';
import SearchBox from '../SearchBox';

export default function Header() {
 return (
   <Navbar bg="white" expand="lg" className="border-bottom border-custom-gray">
     <Navbar.Brand className="mx-3">
       <Link href="/restaurants">
         <a>
           <Image
             src="/logo.png"
             alt="OneBitFood"
             width={200}
             height={44}
             className="clickable_effect"
           />
         </a>
       </Link>
     </Navbar.Brand>
     <Navbar.Toggle aria-controls="responsive-navbar-nav" />
     <Navbar.Collapse id="responsive-navbar-nav" className="justify-content-end">
       <SearchBox />
```

```
28        </Navbar.Collapse>
29      </Navbar>
30    )
      }
```

4 – Atualize o getRestaurant.js colocando:

Default

```
1     import useSWR from 'swr';
2     import { useRouter } from 'next/router';
3     const fetcher = (...args) => fetch(...args).then((res) => res.json())
4
5     export default function <span class="md-plain">getRestaurants</span>() {
6      const router = useRouter();
7      const { category, q } = router.query;
8
9      let params = '';
10    if(category)
11      params = `${params == '' ? '?' : '&'}category=${category}`
12    if(q)
13      params = `${params == '' ? '?' : `${params}&`}q=${q}`
14
15    const { data, error } = useSWR(
16      `${process.env.apiUrl}/api/restaurants${params}`,
17      fetcher, { revalidateOnFocus: false }
18    )
19
20    return { restaurants: data, isLoading: !error && !data, isError: error }
21    }
```

5 – Teste o APP (teste na versão mobile também):

Default

```
yarn dev
```

14 – Tela de detalhes do Restaurante

Nesta aula vamos criar a tela de visualização do restaurante com os pratos que ele possui

1- Crie uma page para mostrar os detalhes do restaurante em `pages/restaurants`, crie um arquivo chamado [id].js

2 – Crie um component para colocar os detalhes do restaurante chamado DetailsRestaurant:

Default

```
mkdir components/DetailsRestaurant
touch components/DetailsRestaurant/index.js
```

3 – Na página [id].js coloque a estrutura da página:

Default

```
import DetailsRestaurant from "../../components/DetailsRestaurant";

export default function Restaurant() {
 return <DetailsRestaurant />;
}
```

4 – Para dividirmos melhor a tela de detalhes crie os seguintes components:

Default

```
mkdir components/DetailsRestaurant/Details
touch components/DetailsRestaurant/Details/index.js
mkdir components/DetailsRestaurant/CategoryProducts
touch components/DetailsRestaurant/CategoryProducts/index.js
```

5 – Crie a base do component DetailsRestaurant/index.js colocando:

Default

```
import React from 'react';
import Container from 'react-bootstrap/Container';
import Details from './Details';
import CategoryProducts from './CategoryProducts';

export default function DetailsRestaurant(props) {
 return(
   <>
     <Details/>
     <CategoryProducts/>
   </>
)
}
```

6 – Prepare a base dos components que serão usados: a – Details

Default

```
export default function Details(props) {
 return(
   <>
   </>
)
}
```

b – CategoryProducts

Default

```
export default function CategoryProducts(props) {
 return(
   <>
   </>
)
}
```

7 – Vamos criar um service para pegar os detalhes do restaurante, em services crie getRestaurant.js e coloque nele:

Default

```
import useSWR from 'swr';

export default function getRestaurant(id) {
 const fetcher = (...args) => fetch(...args).then((res) => res.json());

 const { data, error } = useSWR( id ? `${process.env.apiUrl}/api/restaurants/${id}` : null,
   fetcher, { revalidateOnFocus: false }
 )

 return { restaurant: data, isLoading: !error && !data, isError: error }
}
```

8 – Vamos baixar os dados do restaurante no component DetailsRestaurant/index.js usando nosso getRestaurant:

Default

```
import Details from './Details';
import CategoryProducts from './CategoryProducts';
import getRestaurant from '../../services/getRestaurant';
import { useRouter } from 'next/router';
import { Spinner, Alert } from 'react-bootstrap';

export default function DetailsRestaurant() {
 const router = useRouter();
 const { id } = router.query;

 const { restaurant, isLoading, isError } = getRestaurant(id);

 if(isError)
   return <Alert variant='custom-red'>Erro ao carregar</Alert>
 else if(isLoading)
   return <Spinner animation='border'/>

 return(
   <>
     <Details {...restaurant} />
    {restaurant.product_categories.map((product_category, i) =>
       <CategoryProducts restaurant={restaurant} {...product_category} key={i} />
    )}
   </>
  )
}
```

9 – No component Details criado, coloque:

Default

```
1  import { Row, Col, Card } from 'react-bootstrap';
2  import Image from 'next/image'
3  import { FaStar } from 'react-icons/fa';
4  import toCurrency from '../../../services/toCurrency';
5
6  export default function Details(props) {
7    return (
8      <>
9        <h3 className='fw-bold'>{props.name}</h3>
10       <Card className="mt-2 mb-4">
11         <Row className="my-3 mx-1">
12           <Col md={3} >
13             <Image
14               src={props.image_url}
15               alt={props.name}
16               width={300}
17               height={200}
18               layout="responsive"
19             />
20           </Col>
21           <Col md={9}>
22             <p><small>{props.description}</small></p>
23             <Row className='row-cols-auto'>
24               <Col  className="pr-0">
25                 <small className='border px-3 border-custom-gray fw-bold'>
26                   entrega {toCurrency(props.delivery_tax)}
27                 </small>
28               </Col>
29               <Col >
30                 <small className='fw-bold'>{props.category_title}</small>
```

```
31              </Col>
32              <Col >
33                <span className='text-custom-orange'>
34                  <FaStar/> 5
35                </span>
36              </Col>
37            </Row>
38          </Col>
39        </Row>
40      </Card>
41    </>
42  )
43 }
```

10 – No component CategoryProducts coloque:

Default

```
1   import { Row, Col, Card } from 'react-bootstrap';
2   import Image from 'next/image'
3   import toCurrency from '../../../services/toCurrency';
4   import truncateString from '../../../services/truncateString';
5
6   export default function CategoryProducts(props) {
7
8     return(
9       <>
10        <h5 className='fw-bold'>{props.title}</h5>
11        <Row>
12          {props.products.map((product, i) =>
13            <Col md={4} sm={12} key={i}>
14              <Card className="mb-4 clickable_effect">
```

```
15              <Row className="my-3 mx-1">
16                <Col md={6} xs={{span: 12, order: 2 }}>
17                  <p className='fw-bold mb-0'>{product.name}</p>
18                  <p><small>{truncateString(product.description, 80)}</small></p>
19                  <small className='border px-3 border-custom-gray fw-bold'>
20                    {toCurrency(product.price)}
21                  </small>
22                </Col>
23
24                <Col md={6} xs={{span: 12, order: 1 }} >
25                  <Image
26                    src={product.image_url}
27                    alt={product.name}
28                    width={300}
29                    height={200}
30                    layout="responsive"
31                  />
32                </Col>
33              </Row>
34            </Card>
35          </Col>
36        )}
37      </Row>
38    </>
39  )
40 }
```

15 – Instalando o Recoil

O que é o Recoil?

O Recoil é uma biblioteca criada dentro do Facebook (assim como o próprio React) que tem como objetivo te ajudar a gerenciar estados de uma forma fácil e intuitiva (usar o recoil é como usar um hook qualquer no React).

Entendendo o Recoil

Na versão atual o Recoil é baseado principalmente em dois pilares, os Atoms e os Selectors.

Atoms

Atoms são unidades de estado, é possível atualizar e ler estes estados de uma forma fácil. Também é possível conectar seus components a estes Atoms para que quando eles sejam atualizados os components sejam renderizados novamente. Você também pode usar os Atoms ao invés dos estados locais dos components e utiliza-los para compartilhar os estados entre muitos components.

Selectors

Os Selectors sâo funções puras (que tem como objetivo devolver valores derivados dos Atoms) que recebem um Atom como argumento, quando o Atom que veio como argumento é atualizado o selector também atualiza o valor de retorno. Assim como no caso dos Atoms, os Components também podem se 'inscrever' para serem avisados quando os selectors forem atualizados, quando isso acontece eles são renderizados novamente.

1 – Instale o recoil + recoil persist rodando:

Default

```
1  yarn add recoil recoil-persist
```

2 – No arquivo _app.js:

a – importe o recoil:

Default

```
1  import { RecoilRoot } from 'recoil';
```

b – Em volta de <Component {...pageProps} /> coloque:

Default

```
1  <RecoilRoot>
2   <Component {...pageProps} />
3  </RecoilRoot>
```

c – Crie as seguintes pastas:

Default

```
1  mkdir store
2  mkdir store/atoms
3  touch store/atoms/addressAtom.js
```

d – No atom address criado coloque:

Default

```
import { atom } from 'recoil';
import { recoilPersist } from 'recoil-persist'

const { persistAtom } = recoilPersist()

const addressState = atom({
 key: 'addressState',
 default: {
     city: "",
     neighborhood: "",
     street: "",
     number: "",
     complement: ""
  },
 effects_UNSTABLE: [persistAtom]
});

export default addressState;
```

16 – Desafio friendly id

1 – Instale o friendly id na API e configure ele nos Restaurantes

2 – Devolva o slug do restaurante junto com os outros detalhes dele em /restaurants e /restaurants/:id

3 – Altere o cliente web para usar o friendly id na url e na chamada da API