

OneBitFood V2 – Aula 2

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

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-96Xa4I8AcdsuFcov_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

```

1  import Head from 'next/head';
2  import '../styles/globals.scss';
3
4  export default function MyApp({ Component, pageProps }) {
5    return(
6      <>
7        <Head>
8          <title>OneBitFood V2</title>
9        </Head>
10
11        <main>
12          <Component {...pageProps} />
13        </main>
14      </>
15    )
16  }

```

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

Default

```

1  export default function Home() {
2    return (
3      <>
4      </>
5    )
6  }

```

Assim teremos um código menos verboso nesse início 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 início 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

1 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

```
1  import Head from 'next/head';
2  import './styles/globals.scss';
3
4  export default function MyApp({ Component, pageProps }) {
5    return(
6      <>
7        <Head>
8          <title>OneBitFood V2</title>
9          <link rel="icon" href="/favicon.ico" />
10         </Head>
11
12        <main>
13          <Component {...pageProps} />
14        </main>
15      </>
16    )
17  }
```

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

Default

```
1 $custom-red: #FF0043;
2 $custom-gray: #cccccc;
3 $custom-gray-darker: #666666;
4 $custom-black: #353535;
5 $custom-orange: #F7C97C;
```

6 – Em globals.scss coloque:

Default

```
1 @import "colors.scss";
2
3 $theme-colors: (
4   "custom-red": $custom-red,
5   "custom-gray": $custom-gray,
6   "custom-gray-darker": $custom-gray-darker,
7   "custom-black": $custom-black,
8   "custom-orange": $custom-orange
9 );
```

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

```
1  $spacer: 1rem;
2  $spacers: (
3    0: 0,
4    1: ($spacer * .25),
5    2: ($spacer * .5),
6    3: $spacer,
7    4: ($spacer * 1.5),
8    5: ($spacer * 3),
9    6: ($spacer * 5),
10   7: ($spacer * 10),
11   8: ($spacer * 15),
12   9: ($spacer * 20)
13 );
14
15 $font-weight-bold: 600;
```

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

Default

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

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

Default

```

1  html, body {
2    margin: 0px;
3    padding: 0px;
4    height: 100%;
5  }
6
7  body {
8    background-image: url('../public/bg.png');
9  }

```

| 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

```

1  @import url('https://fonts.googleapis.com/css2?
    family=Lexend:wght@300;400;500;600;700&display=swap');
2
3
4  html, body {
5    margin: 0px;
6    padding: 0px;
7    height: 100%;
8    font-family: 'Lexend', sans-serif;
9  }

```

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

```

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

```

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

Default

```
1  import Container from 'react-bootstrap/Container';
2
3  export default function Home() {
4    return (
5      <>
6        <Container>
7          Home Page Content
8        </Container>
9      </>
10   )
11 }
```

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

Default

```
1  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

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

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

Default

```

1  import React from 'react';
2
3  import { Navbar } from 'react-bootstrap';
4  import Image from 'next/image';
5  import Link from 'next/link';
6
7  const Header = () => {
8    return (
9      <Navbar bg="white" expand="lg" className="border-bottom border-custom-
10     gray">
11        <Navbar.Brand>
12          <Link href="/restaurants">
13            <a>
14              <Image
15                src="/logo.png"
16                alt="OneBitFood"
17                width={200}
18                height={44}
19              />
20            </a>
21          </Link>
22        </Navbar.Brand>
23      </Navbar>
24    )
25  }
26
27  export default Header;

```

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

Default

```
1  import Head from 'next/head';
2  import './styles/globals.scss';
3  import Header from "../components/Header";
4  import Container from 'react-bootstrap/Container';
5
6  export default function MyApp({ Component, pageProps }) {
7    return(
8      <>
9        <Head>
10         <title>OneBitFood V2</title>
11         <link rel="icon" href="/favicon.ico" />
12       </Head>
13
14       <main>
15         <Header />
16         <Container className="mt-5">
17           <Component {...pageProps} />
18         </Container>
19       </main>
20     </>
21   )
22 }
```

4 – Verifique com ficou subindo o projeto:

Default

```
1  yarn dev
```

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

a – No globals.scss coloque:

Default

```
1 .clickable_effect:hover {  
2   opacity: 0.8;  
3   transform:scale(1.01);  
4   cursor: pointer;  
5 }
```

b – Na imagem do Header inclua a classe:

Default

```
1 <Image  
2   src="/logo.png"  
3   alt="OneBitFood"  
4   width={200}  
5   height={44}  
6   className="clickable_effect"  
7 />
```

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

```

1  import { Button, Row, Col } from 'react-bootstrap';
2  import Link from 'next/link';
3  import { FaCrosshairs } from 'react-icons/fa';
4
5  export default function Home() {
6    return (
7      <Row className="mt-8 justify-content-center">
8        <Col md={7} xs={12} className="text-center">
9          <h1 className='fw-bolder text-custom-gray-darker mb-5 lh-base display-5'>
10            Comida saudável e gostosa direto na sua casa
11          </h1>
12          <Link href='/restaurants'>
13            <Button variant="custom-red" size="lg" className='text-white'>
14              <FaCrosshairs className='pb-1' />
15              <span className='px-2 fw-bolder'>ENCONTRAR AGORA</span>
16            </Button>
17          </Link>
18        </Col>
19      </Row>
20    )
21  }

```

7 – Efeito de digitação

Vamos incluir na nossa Home um efeito de “digitando” para tornar a experiência 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

```

1  import React, { useState, useEffect } from 'react';
2  ...
3
4  export default function Typewriter(props) {
5    ...
6
7    useEffect(() => {
8      let currentText = "";
9      props.text.split("").forEach((char, index) => {
10        setTimeout( () => {
11          currentText = currentText.slice(0, -1)
12          currentText += char;
13          if(props.text.length !== index + 1) currentText += "I"
14          setPhrase(currentText)
15        }, 200 + (index * 100));
16      })
17    }, []);

```

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

Default

```

1  import { FaCrosshairs } from 'react-icons/fa';
2  import { Container, Button, Row, Col } from 'react-bootstrap';
3  import Link from 'next/link';
4  import Typewriter from '../components/Typewriter';
5
6  export default function Home() {
7    return (
8      <Row className="mt-8 justify-content-md-center">
9        <Col md={7} xs={12} className="text-center">
10         <h1 className="fw-bolder text-custom-gray-darker mb-5 lh-base display-5">
11           <Typewriter text="Comida saudável e gostosa direto na sua casa"/>
12         </h1>
13         <Link href="/restaurants">
14           <Button variant="custom-red" size="lg" className="text-white">
15             <FaCrosshairs/>
16             <span className="px-2">ENCONTRAR AGORA</span>
17           </Button>
18         </Link>
19       </Col>
20     </Row>
21   )
22 }

```

8 – Listando Restaurantes

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

Default

```

1  mkdir pages/restaurants
2  touch pages/restaurants/index.js

```


2 – Crie a estrutura da página:

Default

```
1  export default function Restaurants() {  
2    return (  
3      <>  
4  
5      </>  
6    )  
7  }
```

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

Default

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

4 – Adicione o seguinte conteúdo no component criado:

Default

```

1  import React from 'react';
2  import { Container, Row, Col } from 'react-bootstrap';
3
4  export default function ListRestaurants() {
5
6    return (
7      <div className='mt-5'>
8        <h3 className='fw-bold'>Restaurantes</h3>
9        <Row>
10         <Col>Restaurants...</Col>
11       </Row>
12     </div>
13   )
14 }

```

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

Default

```

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

```

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

a – Crie uma pasta chamada services, rodando:

Default

```
1 mkdir services
```

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

Default

```
1 export default function toCurrency(value) {  
2   const formatter = new Intl.NumberFormat('pt-BR', {  
3     style: 'currency',  
4     currency: 'BRL',  
5   })  
6   return formatter.format(value)  
7 }
```

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

Default

```
1 export default function truncateString(str, num) {  
2   if (str.length <= num)  
3     return str  
4   else  
5     return str.slice(0, num) + '...'  
6 }
```

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

Default

```
1 mkdir components/ListRestaurants/Restaurants  
2 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
36     Ipsum, Javascript Ipsum e Javascript Ipsum',
37     'image_url': '/restaurant.jpeg',
```

```

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

```

1  module.exports = {
2    images: {
3      domains: ['localhost'],
4    },
5    env: {
6      apiUrl: 'http://localhost:3000',
7    },
8  }

```

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

```

1  import useSWR from 'swr';
2
3  export default function getRestaurants() {
4    const fetcher = (...args) => fetch(...args).then((res) => res.json());
5
6    const { data, error } = useSWR(
7      `${process.env.apiUrl}/api/restaurants`,
8      fetcher,
9      { revalidateOnFocus: false }
10   )
11
12   return { restaurants: data, isLoading: !error && !data, isError: error }
13 }

```

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 '../services/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.length == 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 restaurante.
- 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

- 1 `mkdir components/Categories`
- 2 `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

- 1 `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

- 1 `export default {`
- 2 `speed: 500,`
- 3 `slidesToShow: 5,`
- 4 `slidesToScroll: 4,`
- 5 `initialSlide: 0,`
- 6 `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

```

1  import Link from 'next/link';
2  import Image from 'next/image';
3
4  export default function Category(props) {
5    return(
6      <div>
7        <Link href={` /restaurants?category=${props.title}`}>
8          <a>
9            <Image
10              src={props.image_url}
11              alt={props.title}
12              width={300}
13              height={200}
14              className='px-1 clickable_effect'
15            />
16          </a>
17        </Link>
18        <p className='fw-bold'>{props.title}</p>
19      </div>
20    )
21  }

```

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

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   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 '../services/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

```

1  function renderContent() {
2      if(isError)
3          return <Alert variant='custom-red'>Erro ao carregar</Alert>;
4      else if(isLoading)
5          return <Spinner animation="border" />;
6      else {
7          if(categories.length == 0)
8              return <p>Nenhuma categoria disponível ainda</p>;
9          else {
10             return (
11                 <Card className="mt-2">
12                     <Slider {...slickSettings} className="px-4 pt-4 text-center">
13                         {categories.map((category, i) => <Category category={category} key={i}/>)}
14                     </Slider>
15                 </Card>
16             )
17         }
18     }
19 }

```

e – Atualize as importações dos elementos do bootstrap:

Default

```

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

```

f – Atualize o conteúdo do método return:

Default

```
1  return (  
2    <>  
3    <h3 className='fw-bold'>Categorias</h3>  
4    {renderContent()}  
5    </>  
6  )
```

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

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 } = router.query;
8
9    let params = "";
10   if(category)
11     params = `${params == " ? '?' : '&'}category=${category}`
12
13   const { data, error } = useSWR(
14     `${process.env.apiUrl}/api/restaurants${params}`,
15     fetcher, {
16       revalidateOnFocus: false
17     }
18   )
19
20   return { restaurants: data, isLoading: !error && !data, isError: error }
21 }

```

4 – Teste o APP:

Default

```
1 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

```
1  import React from 'react';
2
3  import { Navbar, Nav } from 'react-bootstrap';
4  import Image from 'next/image';
5  import Link from 'next/link';
6  import SearchBox from '../SearchBox';
7
8  export default function Header() {
9    return (
10     <Navbar bg="white" expand="lg" className="border-bottom border-custom-
11     gray">
12       <Navbar.Brand className="mx-3">
13         <Link href="/restaurants">
14           <a>
15             <Image
16               src="/logo.png"
17               alt="OneBitFood"
18               width={200}
19               height={44}
20               className="clickable_effect"
21             />
22           </a>
23         </Link>
24       </Navbar.Brand>
25       <Navbar.Toggle aria-controls="responsive-navbar-nav" />
26       <Navbar.Collapse id="responsive-navbar-nav" className="justify-content-
27       end">
28         <SearchBox />
29       </Navbar.Collapse>
30     </Navbar>
31   );
32 }
```

```

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

```
1 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

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

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

Default

```
1 import DetailsRestaurant from "../../components/DetailsRestaurant";
2
3 export default function Restaurant() {
4   return <DetailsRestaurant />;
5 }
```

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

Default

```
1 mkdir components/DetailsRestaurant/Details
2 touch components/DetailsRestaurant/Details/index.js
3 mkdir components/DetailsRestaurant/CategoryProducts
4 touch components/DetailsRestaurant/CategoryProducts/index.js
```

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

Default

```
1  import React from 'react';
2  import Container from 'react-bootstrap/Container';
3  import Details from './Details';
4  import CategoryProducts from './CategoryProducts';
5
6  export default function DetailsRestaurant(props) {
7    return(
8      <>
9        <Details/>
10       <CategoryProducts/>
11     </>
12   )
13 }
```

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

Default

```
1  export default function Details(props) {
2    return(
3      <>
4        </>
5      )
6  }
```

b – CategoryProducts

Default

```

1  export default function CategoryProducts(props) {
2    return(
3      <>
4      </>
5    )
6  }

```

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

Default

```

1  import useSWR from 'swr';
2
3  export default function getRestaurant(id) {
4    const fetcher = (...args) => fetch(...args).then((res) => res.json());
5
6    const { data, error } = useSWR( id ? `${process.env.apiUrl}/api/restaurants/${id}` :
7      null,
8      fetcher, { revalidateOnFocus: false }
9    )
10
11    return { restaurant: data, isLoading: !error && !data, isError: error }
12  }

```

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

Default

```

1  import Details from './Details';
2  import CategoryProducts from './CategoryProducts';
3  import getRestaurant from '../services/getRestaurant';
4  import { useRouter } from 'next/router';
5  import { Spinner, Alert } from 'react-bootstrap';
6
7  export default function DetailsRestaurant() {
8    const router = useRouter();
9    const { id } = router.query;
10
11    const { restaurant, isLoading, isError } = getRestaurant(id);
12
13    if(isError)
14      return <Alert variant='custom-red'>Erro ao carregar</Alert>
15    else if(isLoading)
16      return <Spinner animation='border' />
17
18    return(
19      <>
20        <Details {...restaurant} />
21        {restaurant.product_categories.map((product_category, i) =>
22          <CategoryProducts restaurant={restaurant} {...product_category} key={i} />
23        )}
24      </>
25    )
26  }

```

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 colouque:

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 são 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

```
1  import { atom } from 'recoil';
2  import { recoilPersist } from 'recoil-persist'
3
4  const { persistAtom } = recoilPersist()
5
6  const addressState = atom({
7    key: 'addressState',
8    default: {
9      city: '',
10     neighborhood: '',
11     street: '',
12     number: '',
13     complement: ''
14   },
15   effects_UNSTABLE: [persistAtom]
16 });
17
18 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