Diferenciais (Bonus) - React e .NET

React.

- Componente no **React** é uma parte da aplicação totalmente independente do restante que representa uma fatia da aplicação.
- Um dos principais pontos é a sua reutilização em diversos locais na aplicação e sua reatividade e comportamentos conforme a ação do usuário
- Os componentes podem receber **props** que são parâmetros que podem reagir com o componente em si, internamente, mudando assim seu comportamento e estado.
- **Estados, eventos** e **efeitos** causados por uma reação de **props** em um componente podem agilizar, facilitar, e modularizar a criação de aplicações **Web**.

Exemplo de uma listagem simples em React.

- **App.jsx**

```
// App
import { useState } from 'react'
import './App.css'
import Fruit from './components/Fruit'

function App() {
  // uso do estado do React
  const [fruits, setFruits] = useState(
    [
      {id: 0, name: "banana"},
      {id: 1, name: "uva"},
```

```
      {id: 2, name: "maçã"},
      {id: 3, name: "pêra"},
      {id: 4, name: "melancia"},
    ]
  )

  const removeFruits = (id) => {
    // remove a fruta pelo id
    const filteredFruits = fruits.filter((fruit) =>{
      return fruit.id !== id
    })

    // muda o estado de fruits
    setFruits(filteredFruits)
  }

  return (
    <>
      {/* Componente Fruta */}
      <Fruit
        // data e função removeFruits como props
        data={fruits}
        removeFruits={removeFruits}
      />
    </>
  )
}

export default App
```

- Componente **FruitsList**

```
// Componete Fruit
const FruitList = ({ data, removeFruits }) => {
```

```
    const fruits = data

    // funcão handleFruits
    const handleFruits = (e, id) => {
        e.preventDefault()

        // devolve o props passando o id
        removeFruits(id)
    }

    return (
        <>
            {fruits.map(fruit => (
                <div key={fruit.id}>
                    <h1>{fruit.name}</h1>
                    <button
                        // evento clique que chama a função: har
                        onClick={
                            (e) => handleFruits(e, fruit.id)
                        }
                    >
                        Remove Fruit
                    </button>
                </div>
            ))}
        </>
    )
}

export default FruitList
```

- Temos a array de objetos fruta, que contem um estado inicial.

```
// App
// uso do estado do React
const [fruits, setFruits] = useState(
  [
    {id: 0, name: "banana"},
    {id: 1, name: "uva"},
    {id: 2, name: "maçã"},
    {id: 3, name: "pêra"},
    {id: 4, name: "melancia"},
  ]
)
```

- Temos o componente sendo renderizado no App
- O componente passa como props, as frutas (data) e a função **removeFruits**

```
// App

return (
  <>
    {/* Componente Fruta */}
    <Fruit
      // data e função removeFruits como props
      data={fruits}
      removeFruits={removeFruits}
    />
  </>
)
```

- No componente **FruitsList** pegamos o props e renderizamos com o map.

```
// FruitsList
// Componete Fruit
```

```
const Fruit = ({ data, removeFruits }) => {
const fruits = data
```

```
// FruitsList
return (
    <>
        {fruits.map(fruit => (
            <div key={fruit.id}>
                <h1>{fruit.name}</h1>
                <button
                    // evento clique que chama a função: handleF
                    onClick={
                        (e) => handleFruits(e, fruit.id)
                    }
                >
                        Remove Fruit
                </button>
            </div>
        ))}
    </>
)
```

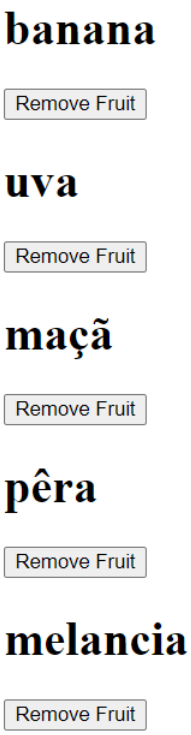
- ao clicar no botão, o evento reage e chama a **handleFruits**, que ativa o **props removeFruits** no componente App passando o id.

```
// FruitsList
// funcão handleFruits
const handleFruits = (e, id) => {
    e.preventDefault()

    // devolve o props passando o id
    removeFruits(id)
}
```

- a função **removeFruits** é chamada, a fruta removida pelo id, e o novo estado é atualizado.

```
// FruitsList
const removeFruits = (id) => {
  // remove a fruta pelo id
  const filteredFruits = fruits.filter((fruit) =>{
    return fruit.id !== id
  })

  // muda o estado de fruits
  setFruits(filteredFruits)
}
```

- **FruitList** com estado inicial



- após clicar no botão referente a melancia:

banana

Remove Fruit

uva

Remove Fruit

maçã

Remove Fruit

pêra

Remove Fruit

.NET

- Posso dizer que estou familiarizado com o .NET pois na minha maior experiência como Desenvolvedor, o Back End era todo feito em C# com o .NET.
- Vantagens do **.NET** para o desenvolvimento **Web**:
    - **Robusto:** o **.NET** é uma tecnologia bem robusta em relação as outras, mantida pela Microsoft, ela está em constante melhoria, seus updates são bem frequentes e ela tem uma integração quase infinita com diversas tecnologias.
    - **Tipagem forte:** Sua tipagem é extremamente forte, deixando assim sua leitura de código mais completa e sujeita a menos erros de compilação.
    - **Orientação de Objetos:** Outra vantagem é a sua orientação a objetos, que é muito completa, segundo velhos colegas de trabalho, o **C#** é uma das melhores linguagens pra isso, junto com o **Java**.
    - **Integração com Web: HTML, CSS e Javascript** podem estar nas views no mesmo projeto que as Controllers e Serviços em **C#**. Assim como o Django, a integração das linguagens Web ficam bem próximas e integradas.
    - **Partial Views:** Partial Views permitem criar classes separadas dentro de um projeto, o que ajuda na manutenção e leitura do código.
    - **Entity Framework:** Poderoso framework que permite que mapear dos elementos de bases de dados para os elementos da aplicação orientada a objetos.
    - **Visual Studio:** Poderoso editor que código da Microsoft que auxilia no Desenvolvimento de aplicações em **C#** e **.NET**.

- **Desvantagens do .NET**
    - **Pesado:** Requer máquinas mais robustas para rodar o Visual Studio da Microsoft e aproveitar o melhor da ferramenta

- **Sintaxe longa:** A sintaxe é extremamente longa e de difícil compreensão principalmente pra quem está querendo aprender.