"Apostila" de React

[React e seus Componentes](https://codesandbox.io/s/prog4-aula-1-qz5md)

Criamos um projeto [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) com o [codesandbox](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67074" \o "CodeSandbox) e analisamos a estrutura do projeto gerada pelo create-[react](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076)-app, como a biblioteca [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) organiza e executa o código do projeto e como podemos definir o que será renderizado através da criação de componentes e importação deles no App.js. Todo componente deve ser uma classe baseada em [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React).Component ou função que exporta/retorna um código jsx (no método render() da classe ou no retorno da função), para que o [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) possa identificar os elementos a serem renderizados em html. É possível exportar diversas funções ou classes de um mesmo arquivo, mas estas devem ser importadas no App.js através da [desestruturação](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment).

[Componentes com Props](https://codesandbox.io/s/prog4-aula-2-4cito)

Os componentes começam a fazer sentido (especialmente na ideia de reuso) quando é possível enviar informações do nosso aplicativo para cada componente, permitindo que sejam utilizados elementos comuns e personalizados os específicos. Ao criar um componente definimos que ele utilizará propriedades como parâmetro (no construtor no caso de uma classe), que funcionam como atributos no html com a diferença de serem totalmente personalizados. O componente pode, então, acessar o objeto props e seus atributos na definição da renderização. Se durante a renderização for utilizado uma propriedade não definida, é possível que o site não seja renderizado (runtime error). Também é possível utilizar a desestruturação para acessar todas as propriedades de uma só vez, de modo que se utilizarmos os mesmos códigos de um atributo html válido para determinado elemento esses podem ser aproveitados sem a necessidade de serem explicitados.

[Componentes (funcionais) com estado](https://codesandbox.io/s/prog4-aula-3-un4kg)

Para que o [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) saiba o que deve ser (re)desenhado na tela, é necessário fornecer uma informação especial, que é definir um objeto state na classe que define o componente. Quando se utiliza a forma funcional dos componentes, é possível utilizar um Hook do [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) (useState) para registrar uma variável de estado e sua forma de atualização.

const [valor , setValor] = useState ("valor inicial que pode ser de qualquer tipo");

Ao utilizar o método que altera o conteúdo da variável 'valor' (e não mudar a variável em uma atribuição simples) o [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) sabe que deve atualizar visualmente esse componente com o novo conteúdo dessa variável/estado. O exemplo utilizado na aula foi um componente que contém a contagem de vezes na qual um botão é clicado, que também foi alterado para compartilhar a informação com o componente pai.

[Roteamento](https://codesandbox.io/s/prog4-aula-5-router-v6-5y7yr?file=/src/components/Content.jsx)

Para fazer a navegação dentro de um projeto com [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) e manter a 'reatividade/reação' que a biblioteca propõe, é necessário manipular uma variável de estado que mantém qual conteúdo está sendo visualizado. Isso impede (ou no mínimo dificulta) a navegação via URL, portanto existem bibliotecas que facilitam esse trabalho. A mais utilizada para roteamento do [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React) na versão web é a [React Router](https://reactrouter.com/). Através dela podemos definir a rota relativa à raiz do projeto que exibe determinado componente. Nessa aula também trabalhamos em conjunto, construindo pedaços do site através de diferentes componentes, mostrando que a ideia de componentização do ambiente facilita muito o trabalho em paralelo dos desenvolvedores.

[Fetch](https://codesandbox.io/s/prog4-aula-06-w5xgj?file=/src/App.js)

Em muitos casos é necessário buscar informações em sistemas externos, que não são parte do mesmo servidor que oferece a aplicação. Nessas situações costuma-se utilizar uma Application Programming Interface (API), que nada mais é que uma forma de um programa acessar dados de outro, ou seja, um servidor que é feito para outros servidores ou aplicativos acessarem. O JavaScript possui diversas bibliotecas com esse intuito (Axios e jQuery são as mais utilizadas para esse fim), mas também possui formas nativas como o XMLHttpRequest e, mais recentemente, o método fetch. Uma forma muito utilizada para comunicar dados entre aplicações é no formato JavaScript Object Notation (JSON), que no caso do JS facilita bastante a conversão das informações recebidas através de uma chamada de API para um objeto disponível no código.

[Storage](https://codesandbox.io/s/prog4-aula-7-i0err)

A [API Storage](https://developer.mozilla.org/pt-BR/docs/Web/API/Storage) permite armazenar informações de sessão ou persistentes de maneira muito simples, uma forma "evoluída" dos cookies que são utilizados para armazenar preferências de sites. A diferença está em como essa informação é tratada: os cookies são transmitidos entre o cliente e o servidor, enquanto os dados armazenados na LocalStorage nunca são enviados ao servidor. No caso do [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React), que de maneira geral realiza sua renderização no cliente, esse modelo de armazenamento é ideal. É possível armazenar chaves e valores (em forma de texto apenas). Além da API da Web, existem diversas bibliotecas, focadas ou não no [React](https://moodle.sertao.ifrs.edu.br/mod/url/view.php?id=67076" \o "React), que se baseiam nela ou em outras soluções de armazenamento para facilitar o desenvolvimento de aplicações que não necessitam de servidores de bancos de dados.

[useEffect](https://codesandbox.io/s/prog4-aula-8-g5q88)

O Hook useEffect() permite definir "efeitos colaterais" a cada vez que um componente é renderizado ou a cada vez que uma dependência definida em sua lista é modificada. Isso permite que ações que devem ser realizadas em conjunto com atualizações na tela (movimentos no banco de dados, mudanças nas configurações do navegador, etc) sejam desassociadas das funções específicas dos componentes, assim como manipular cada efeito isoladamente para todo o ciclo de vida do componente. Ao adicionar um retorno nas funções dentro do useEffect, determinamos o que acontece ao desmontar um componente.