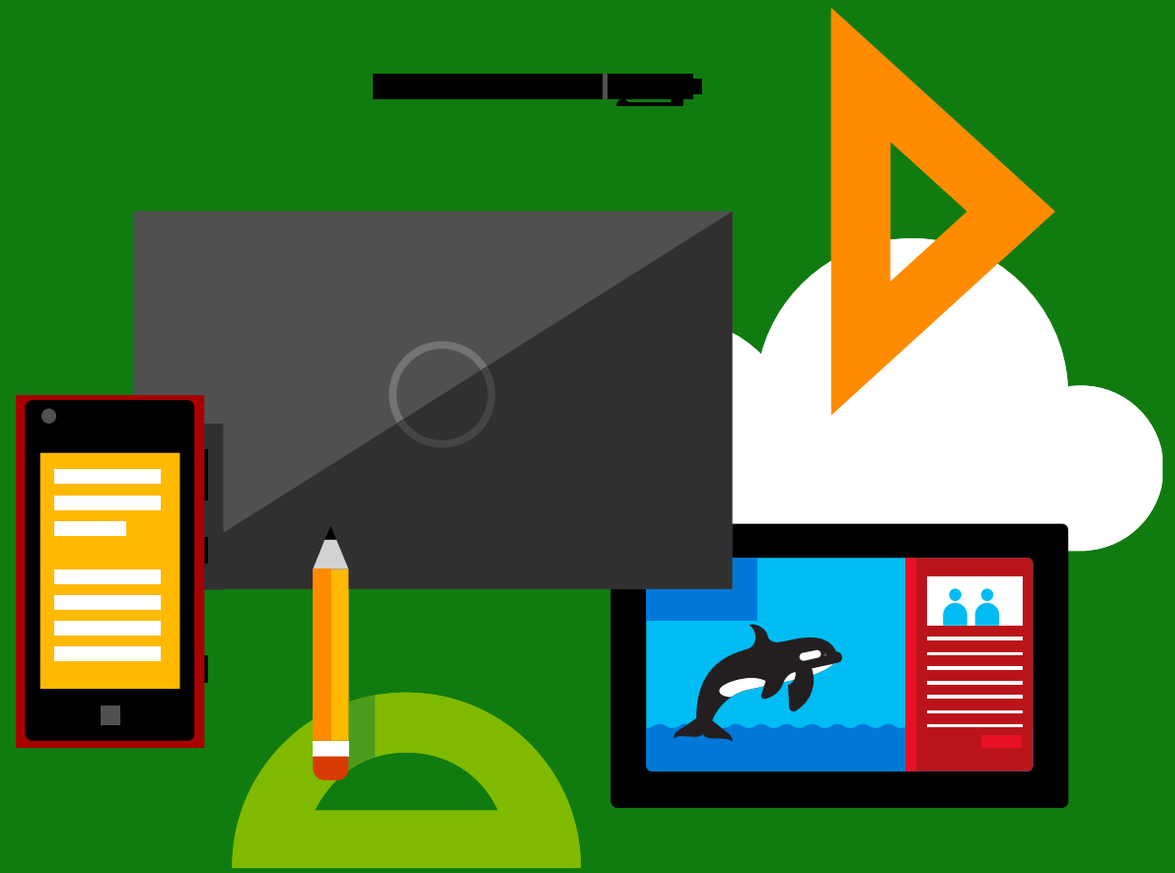
DBStart

React
Instrutor: Júlio Pereira Machado (julio.machado@pucrs.br)

React
Formulários



React – Formulários

- Em um formulário HTML, os elementos naturalmente mantêm algum estado interno
- Por padrão, o estado de um formulário é submetido via HTTP POST pelo evento de submissão disparado por um botão do tipo "submit"
  - Desabilitar POST padrão através do método `event.preventDefault()` no tratador do evento `onSubmit` do `form`
- React pode lidar com formulários de duas formas:
  - Componentes sem controle (*useRef*)
  - Componentes controlados (*useState*)

```
<form>
  <label>
    Name:
    <input type="text" name="name" />
  </label>
  <input type="submit" value="Submit" />
</form>
```

React – Formulários

- Componentes controlados
  - O valor de um elemento é sempre controlado pelo estado definido no componente que gerencia o formulário via React
  - Com implementar:
    - Utilizar o Hook `useState` para controlar o estado
    - Propriedade `value` (ou equivalente, tal como `checked` para checkboxes) associada a um estado
    - Tratador do evento `onChange` para processar entrada de dados e alterar o estado

```
const [formData, setFormData] = useState<FormData>({name:'', jedi: false});
```

```
const handleNameChange: React.ChangeEventHandler<HTMLInputElement> = (event) => {
    setFormData({
        ...formData,
        [event.target.name]: event.target.value
    });
};
```

```
const handleSubmit: React.FormEventHandler<HTMLFormElement> = (event) => {
    event.preventDefault();
    alert(`You submitted the form: ${formData.name} ${formData.jedi}`);
};
```

```
return (
    <>
        <h1>Controlled Form</h1>
        <form onSubmit={handleSubmit}>
            <fieldset>
                <label>
                    <p>Name</p>
                    <input name="name" value={formData.name} onChange={handleNameChange}/>
                </label>
                <label>
                    <p>Jedi</p>
                    <input type="checkbox" name="jedi" checked={formData.jedi} onChange={handleJediChange}/>
                </label>
            </fieldset>
            <button type="submit">Submit</button>
        </form>
    </>
);
```

React – Formulários

- Componentes sem controle
  - O valor de um elemento é gerenciado pelo próprio DOM
  - Como implementar:
    - Utilizar o Hook `useRef` para referenciar diretamente um elemento DOM
    - Importante: alteração do estado de um *ref* não dispara o processo de renderização do React!
    - Obter referência para o elemento DOM através da propriedade `current`

```
const nameInputRef = useRef<HTMLInputElement>(null);
const jediCheckboxRef = useRef<HTMLInputElement>(null);
```

```
const handleSubmit: React.FormEventHandler<HTMLFormElement> = (event) => {
    event.preventDefault();
    alert(`You submitted the form: ${nameInputRef.current?.value} ${jediCheckboxRef.current?.checked}`);
};
```

```
return (
    <>
        <h1>Controlled Form</h1>
        <form onSubmit={handleSubmit}>
            <fieldset>
                <label>
                    <p>Name</p>
                    <input name="name" ref={nameInputRef}/>
                </label>
                <label>
                    <p>Jedi</p>
                    <input type="checkbox" name="jedi" ref={jediCheckboxRef}/>
                </label>
            </fieldset>
            <button type="submit">Submit</button>
        </form>
    </>
);
```

React – Formulários

- O processamento de um formulário pode se tornar bastante complexo pois envolve múltiplas tarefas:
  - Gerenciamento do estado de múltiplos elementos
  - Validação de dados
  - Aplicação de máscaras para entrada de dados
  - Componentes ricos de interface de usuário
  - etc.

React – Formulários

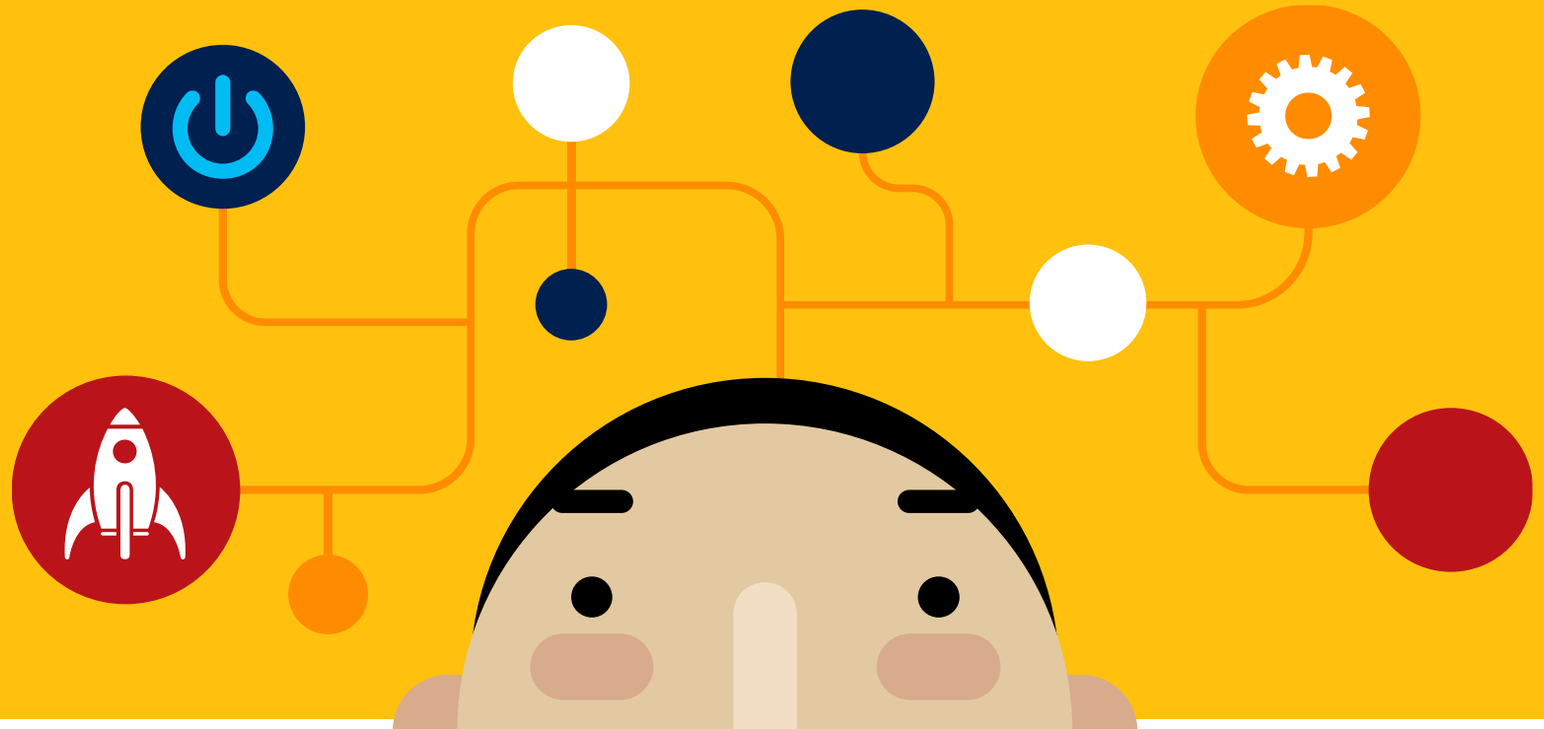
- Bibliotecas de componentes ricos:
  - MUI https://mui.com/
  - React Bootstrap https://react-bootstrap.github.io/
  - Ant Design https://ant.design/
  - PrimeReact https://primereact.org/
  - etc

React – Formulários

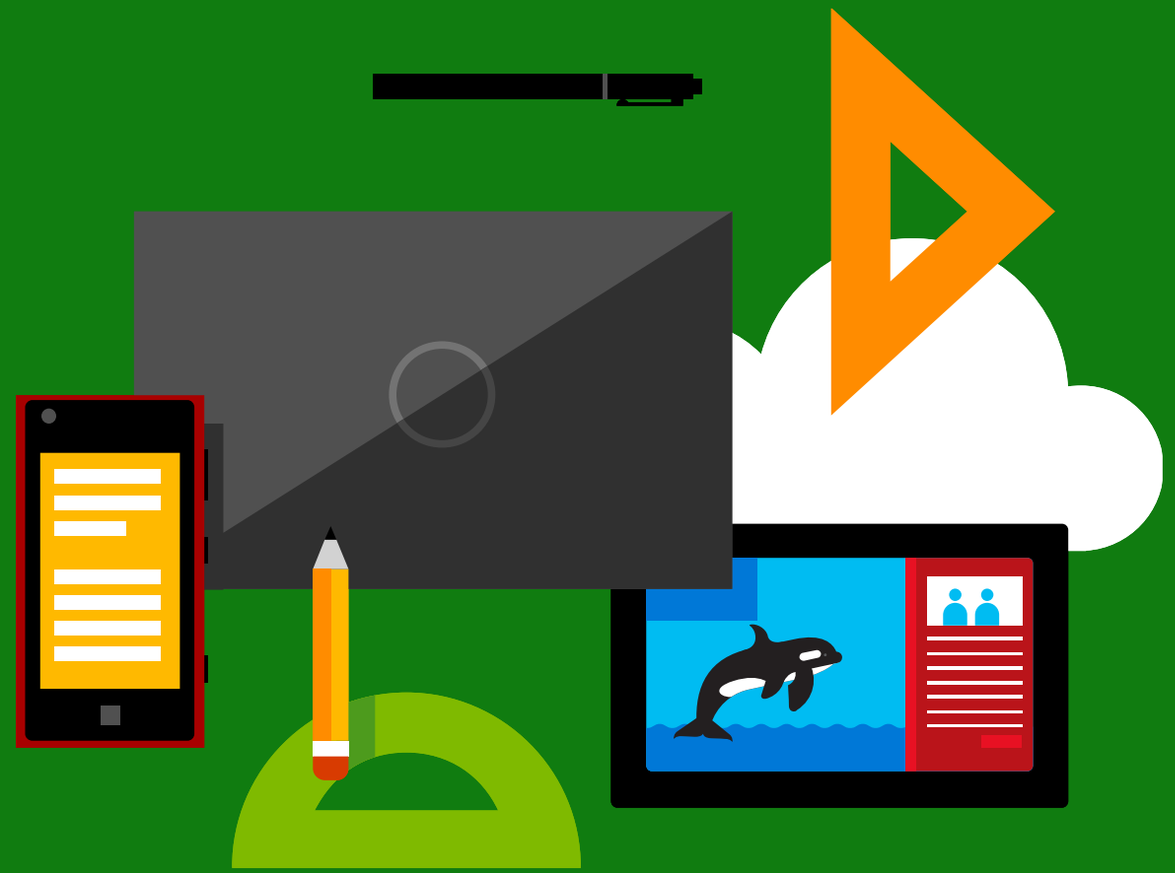
- Bibliotecas para processamento de formulários:
  - Formik https://formik.org/
  - React Hook Form https://react-hook-form.com/
  - React Final Form https://final-form.org/
  - etc

Laboratório

- Siga as instruções do arquivo Lab03_React_Componentes



React
Estado



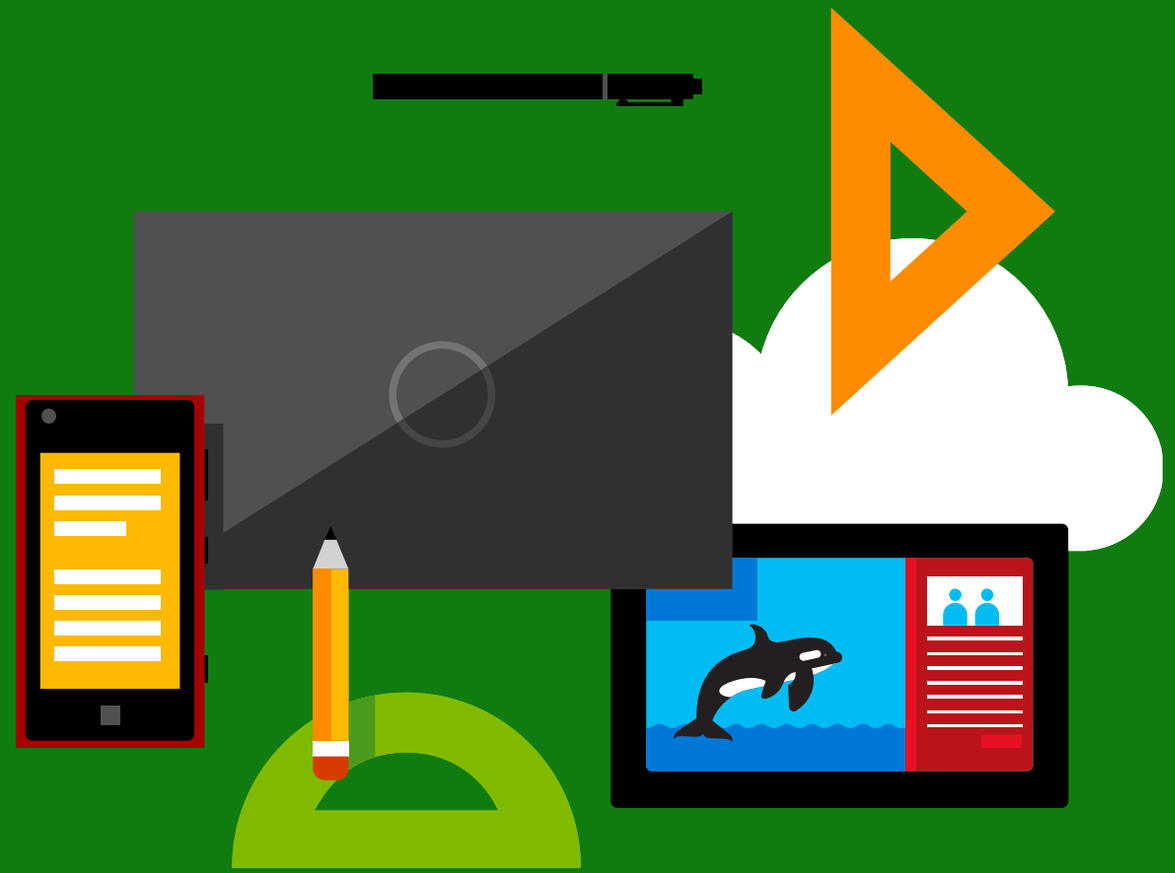
React - Gerenciamento de Estado

- Gerenciamento de dados compartilhado entre os componentes de uma aplicação
  - Estado global da aplicação
  - Estado local do componente
- Estado
  - Persistente (em dispositivo de armazenamento persistente)
  - Temporário (em memória)

React - Gerenciamento de Estado

- Gerenciamento simples de estado de componentes faz uso de React Hooks (useState, useContext, useReducer, etc)
- Gerenciamento complexo de estado da aplicação faz uso de APIs e frameworks adicionais
  - Redux: https://redux.js.org/, https://redux-toolkit.js.org/, https://react-redux.js.org/
  - Recoil: https://recoiljs.org/

Complementos ao React



Complementos

- REACT é uma biblioteca para a criação de componentes de IU
- Suporta o desenvolvimento de aplicações web, mobile e desktop
- Uma aplicação web necessita de funcionalidades adicionais
  - Roteamento
  - Gerenciamento de estado complexo
  - Estilos
  - Client side rendering
  - Server side rendering
  - Universal rendering
  - etc

Complementos

- Diferentes APIs para implementar a navegação entre “páginas”

- React Router
  - https://reactrouter.com/
- Reach Router
  - https://reach.tech/router/
- React Location
  - https://react-location.tanstack.com/

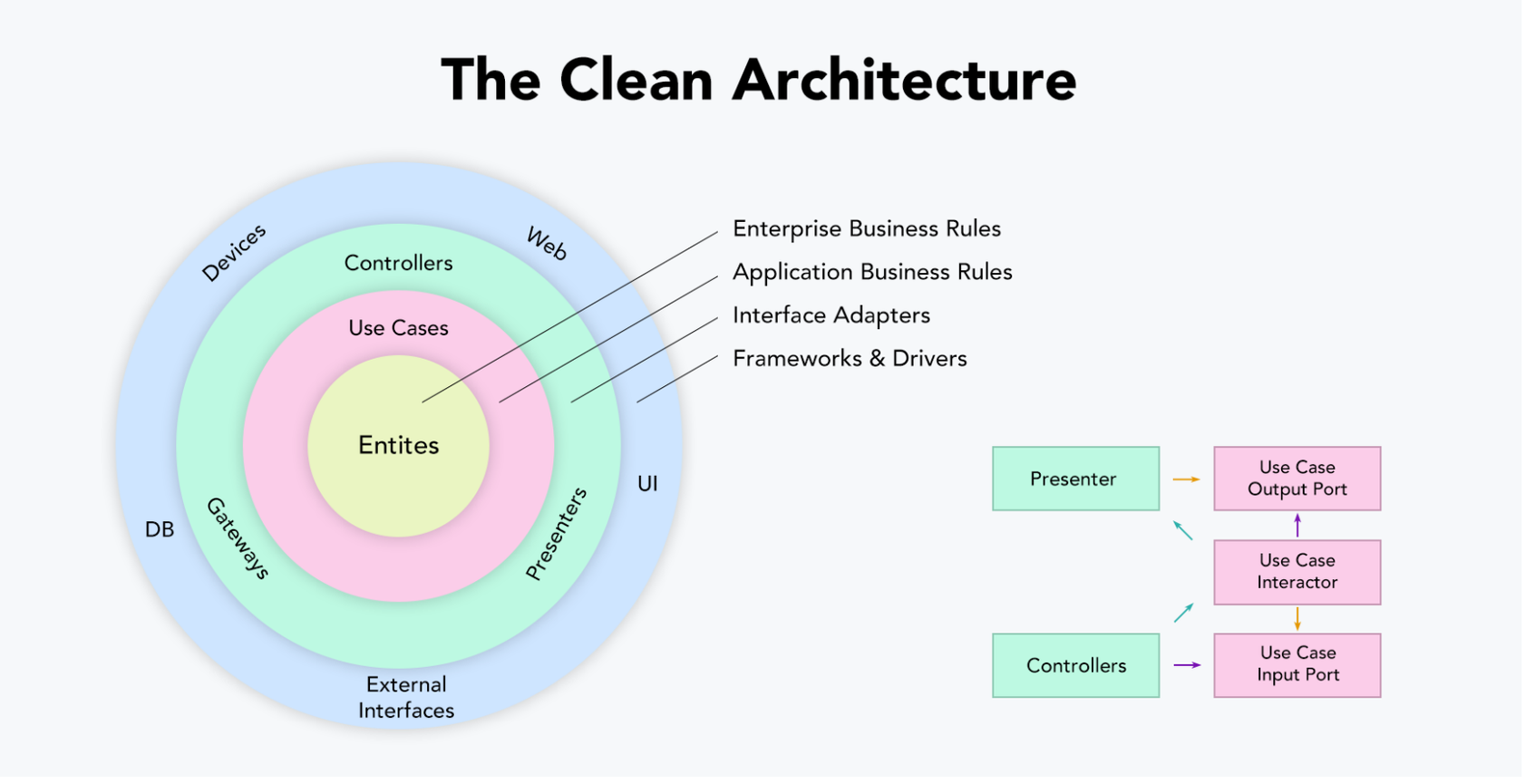
Complementos

- Diferentes frameworks para diferentes necessidades!

- Next.js
  - https://nextjs.org/
- Gatsby
  - https://www.gatsbyjs.com/
- Remix
  - https://remix.run/

Clean Architecture

- React com Clean Architecture
- https://github.com/falsy/react-with-clean-architecture



Estilos

- Artigos comparando diversas formas de gerenciar as folhas e estilo em projetos com React
- https://www.freecodecamp.org/news/how-to-style-a-react-app/

Laboratório

- Siga as instruções do arquivo Lab03_React_Componentes