

FRONT-END DESIGN ENGINEERING

[React – LocalStorage e SessionStorage]

[Aula 20]
[Atualizado em: 11/04/2024]



REACT – LOCALSTORAGE E SESSIONSTORAGE

REACT – LOCALSTORAGE E SESSIONSTORAGE

- ✓ **Introdução ao LocalStorage e SessionStorage**
- ✓ Existem duas possibilidades principais para o armazenamento de dados no navegador: **LocalStorage** e **SessionStorage**.
- ✓ **LocalStorage** e **SessionStorage** são dois recursos fornecidos pelo navegador da web para armazenamento de dados do lado do cliente (ou seja, no navegador do usuário) em aplicações web. Ambos são implementados como parte da *API Web Storage* do **HTML5** e oferecem uma maneira de armazenar dados temporários ou persistentes na máquina do cliente, sem a necessidade de interagir com o servidor.

REACT – LOCALSTORAGE E SESSIONSTORAGE

- ✓ **Introdução ao LocalStorage e SessionStorage**

- ✓ **LocalStorage**

- ✓ É uma forma de armazenamento de dados persistente e que persiste mesmo após o fechamento do navegador. Os dados armazenados no LocalStorage permanecem disponíveis para qualquer aba ou janela do navegador que acesse o mesmo domínio.

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ LocalStorage

- ✓ **Limite de Armazenamento:** A capacidade é em média de 5MB, mas varia de navegador para navegador.
- ✓ **Armazenamento Persistente:** Os dados armazenados no **LocalStorage** permanecem no navegador mesmo após o fechamento e reabertura do navegador, bem como reinicializações do sistema operacional.
- ✓ **Persistência de Dados:** A menos que os dados sejam explicitamente removidos pelo código da aplicação ou pelo usuário, eles permanecem no armazenamento de forma persistente.
- ✓ **Escopo de Domínio:** Os dados do **LocalStorage** estão vinculados ao domínio específico que os criou. Isso significa que eles são acessíveis por qualquer aba ou janela do navegador que esteja acessando o mesmo domínio.

REACT – LOCALSTORAGE E SESSIONSTORAGE

- ✓ **Introdução ao LocalStorage e SessionStorage**

- ✓ **SessionStorage**

- ✓ É semelhante ao LocalStorage, mas os dados armazenados nele são válidos apenas durante a sessão do navegador. Isso significa que os dados são excluídos quando a sessão do navegador é fechada (quando todas as abas do navegador que acessam o mesmo domínio são fechadas).

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ LocalStorage

- ✓ **Limite de Armazenamento:** Tem uma capacidade de armazenamento maior em comparação com o LocalStorage. A capacidade é 5 a 10MB, mas varia de navegador para navegador.
- ✓ **Armazenamento Temporário:** Os dados armazenados no SessionStorage são válidos apenas durante a sessão do navegador. Isso significa que eles são perdidos quando o navegador é fechado.
- ✓ **Tempo de vida limitado:** Os dados no SessionStorage existem apenas enquanto a sessão do navegador está ativa. Quando todas as abas ou janelas que acessam o mesmo domínio são fechadas, os dados são removidos automaticamente.
- ✓ **Persistência de Dados:** A menos que os dados sejam explicitamente removidos pelo código da aplicação ou pelo usuário, eles permanecem no armazenamento de forma persistente.
- ✓ **Escopo de janela:** Os dados do SessionStorage são específicos para cada janela ou aba do navegador. Isso significa que uma aba não pode acessar os dados do SessionStorage criados por outra aba.

REACT – LOCALSTORAGE E SESSIONSTORAGE

- ✓ **Introdução ao LocalStorage e SessionStorage**
- ✓ O ciclo de vida das informações armazenadas no LocalStorage e SessionStorage é determinado pela forma como esses armazenamentos são gerenciados pelo navegador.
- ✓ Em resumo, enquanto o LocalStorage é usado para armazenar dados de forma persistente entre sessões do navegador, o SessionStorage é usado para armazenar dados temporários que são válidos apenas durante a sessão atual do navegador.

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

- ✓ Ambos LocalStorage e SessionStorage em JavaScript oferecem métodos semelhantes para manipulação de dados, pois ambos seguem o padrão de armazenamento chave-valor. Aqui estão os métodos comuns para ambos.

- ✓ **setItem(chave, valor):** Adiciona um item ao armazenamento, com uma chave especificada.

- ✓ `localStorage.setItem('nome', 'João');`

- ✓ **getItem(chave):** Recupera o valor associado a uma chave específica.

- ✓ `const nome = localStorage.getItem('nome');`

- ✓ **removeItem(chave):** Remove o item associado a uma chave específica.

- ✓ `localStorage.removeItem('nome');`

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ **clear():** Remove todos os itens armazenados.

✓ `localStorage.clear();`

✓ **key(index):** Retorna a chave armazenada em um índice específico.

✓ `const primeiraChave = localStorage.key(0);`

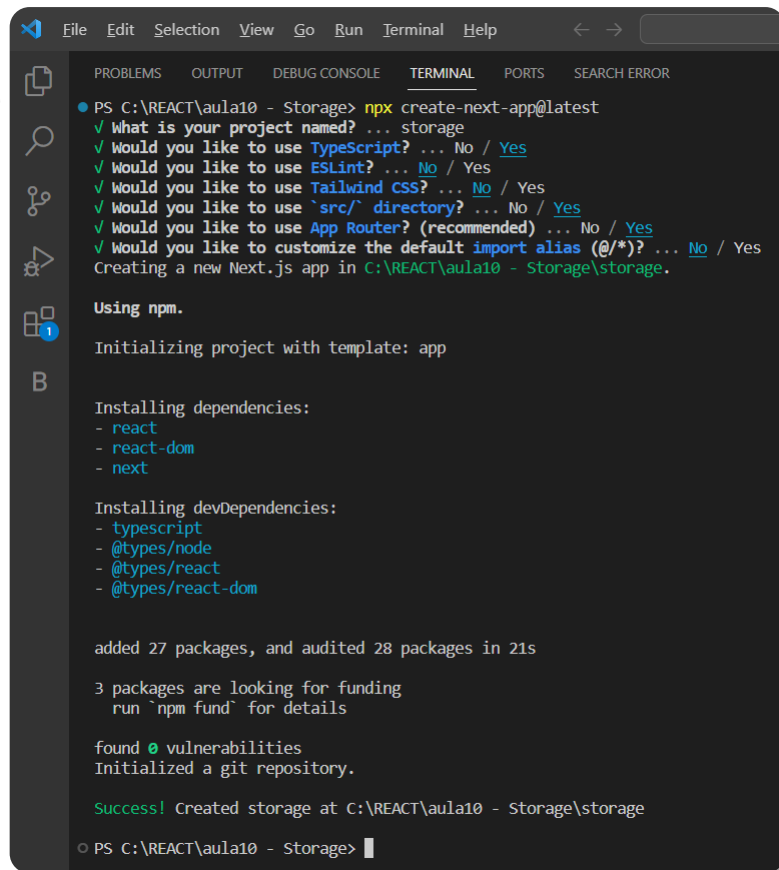
✓ **length:** Retorna o número de itens armazenados no armazenamento.

✓ `const numeroItens = localStorage.length;`

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

- ✓ Vamos criar um novo projeto.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS C:\REACT\aula10 - Storage> npx create-next-app@latest
✓ What is your project named? ... storage
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like to use `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in C:\REACT\aula10 - Storage\storage.

Using npm.

Initializing project with template: app

Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom

added 27 packages, and audited 28 packages in 21s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Initialized a git repository.

Success! Created storage at C:\REACT\aula10 - Storage\storage

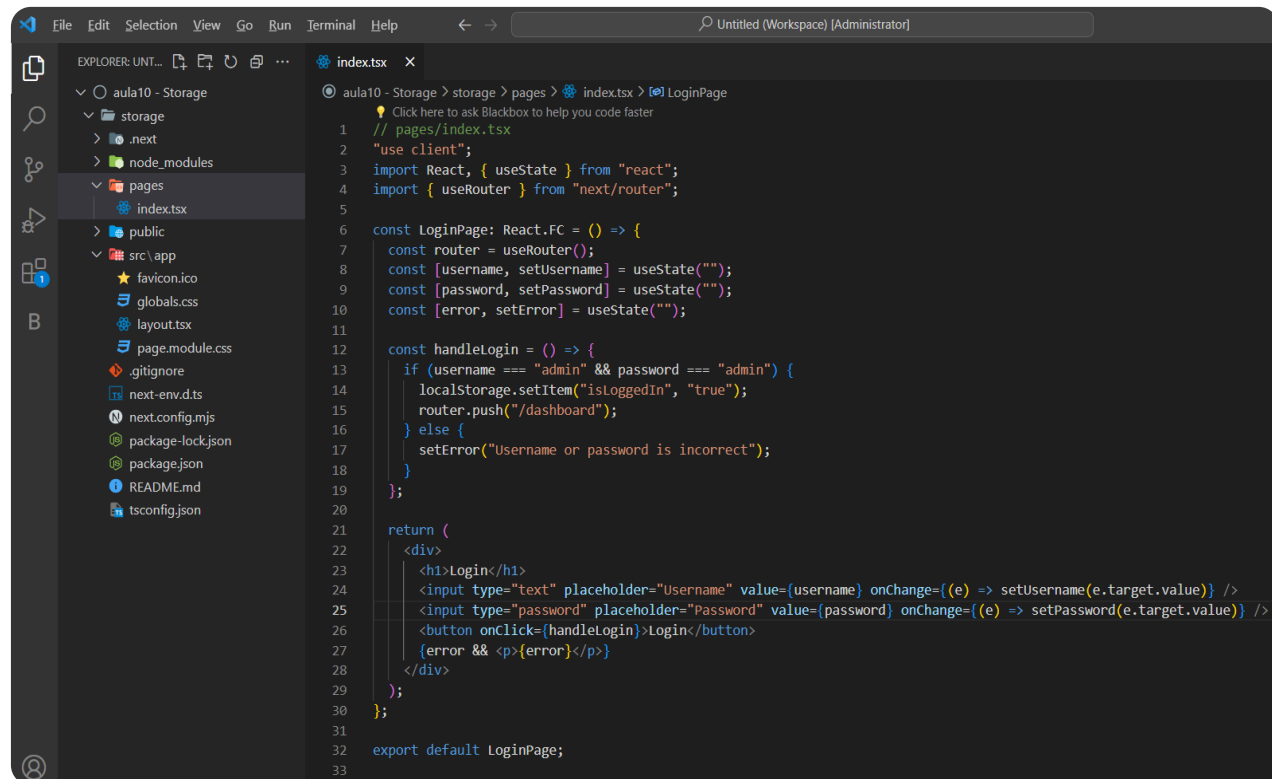
PS C:\REACT\aula10 - Storage>
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

- ✓ **Introdução ao LocalStorage e SessionStorage**
 - ✓ Na sequência vamos criar os seguintes componentes, todos dentro da pasta PAGES:
 - ✓ index.tsx
 - ✓ dashboard.tsx
 - ✓ _app.tsx

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'aula10 - Storage' with folders for 'storage', 'pages', and 'public'. The 'pages' folder is selected, and the 'index.tsx' file is open in the code editor. The code defines a 'LoginPage' component that uses 'useState' for form state and 'useRouter' for navigation. It includes a 'handleLogin' function that checks credentials and uses 'localStorage.setItem' to store a login status. The component renders a form with username and password inputs, a login button, and an error message.

```
1 // Click here to ask Blackbox to help you code faster
2 // pages/index.tsx
3 "use client";
4 import React, { useState } from "react";
5 import { useRouter } from "next/router";
6
7 const LoginPage: React.FC = () => {
8   const router = useRouter();
9   const [username, setUsername] = useState("");
10  const [password, setPassword] = useState("");
11  const [error, setError] = useState("");
12
13  const handleLogin = () => {
14    if (username === "admin" && password === "admin") {
15      localStorage.setItem("isLoggedIn", "true");
16      router.push("/dashboard");
17    } else {
18      setError("Username or password is incorrect");
19    }
20  };
21
22  return (
23    <div>
24      <h1>Login</h1>
25      <input type="text" placeholder="Username" value={username} onChange={e => setUsername(e.target.value)} />
26      <input type="password" placeholder="Password" value={password} onChange={e => setPassword(e.target.value)} />
27      <button onClick={handleLogin}>Login</button>
28      {error && <p>{error}</p>}
29    </div>
30  );
31
32  export default LoginPage;
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exemplo 01 – Analisando linha a linha o arquivo index.tsx

```
1 // pages/index.tsx

  'use client'
  // É uma convenção para indicar que o componente é renderizado no cliente, significando que ele é executado no
  navegador do usuário, após o carregamento inicial da página. Essa convenção ajuda a identificar facilmente
2 quais partes do código são específicas do cliente, em oposição às partes que podem ser executadas no servidor.
  Isso pode ser útil em projetos que utilizam renderização do lado do servidor (SSR) ou que têm lógica
  compartilhada entre o cliente e o servidor.

3 import React, { useState } from "react";

  import { useRouter } from "next/router";
4 // Aqui estamos importando as funções useState e useRouter do React e Next.js, respectivamente. useState é um hook
  do React que permite adicionar estado a componentes funcionais, enquanto useRouter é um hook do Next.js que
  fornece acesso ao objeto de roteamento.

5

6 const LoginPage: React.FC = () => {
  // Aqui estamos definindo um componente funcional chamado LoginPage. Ele usa a notação de tipo do TypeScript para
  indicar que este é um componente funcional React que não recebe props externas.
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exemplo 01 – Analisando linha a linha o arquivo index.tsx

```
7      const router = useRouter();
8
9      const [username, setUsername] = useState("");
10     const [password, setPassword] = useState("");
11
12     const [error, setError] = useState("");
13     //Aqui estamos inicializando o hook useRouter para acessar o objeto de roteamento do Next.js. Em seguida,
14     estamos definindo três estados usando o hook useState: username, password e error. O estado username e
15     password são inicializados como strings vazias, enquanto error é inicializado como uma string vazia.
16
17
18     const handleLogin = () => {
19
20         if (username === "admin" && password === "admin") {
21
22             localStorage.setItem("isLoggedIn", "true");
23
24             router.push("/dashboard");
25
26         } else {
27
28             setError("Username or password is incorrect");
29
30         }
31
32     };
33
34     // Aqui estamos definindo uma função handleLogin que é chamada quando o botão de login é clicado. Ele
35     verifica se o username e password são iguais a "admin" e, se forem, define uma flag no localStorage
36     indicando que o usuário está logado e redireciona para a página de dashboard. Caso contrário, define uma
37     mensagem de erro.
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

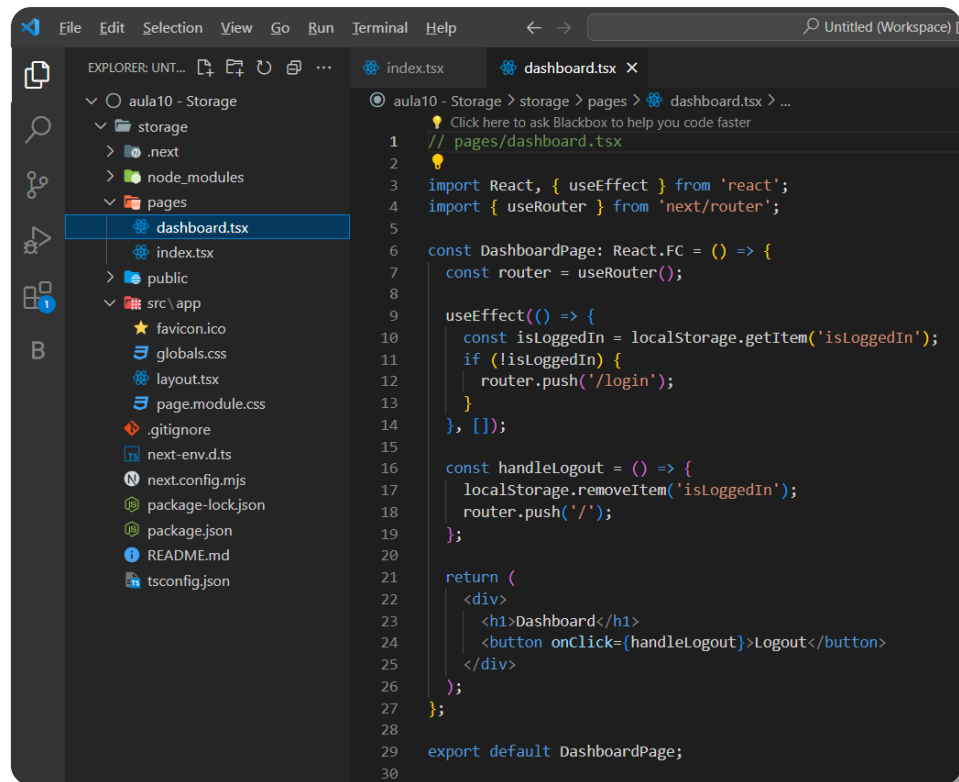
✓ Exemplo 01 – Analisando linha a linha o arquivo index.tsx

```
20
21     return (
22         <div>
23             <h1>Login</h1>
24             <input type="text" placeholder="Username" value={username} onChange={(e) => setUsername(e.target.value)}/>
25             <input type="password" placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)} />
26             <button onClick={handleLogin}>Login</button>
27             {error && <p>{error}</p>}
28         </div>
29     );
30 };
31
32 export default LoginPage
```

```
33 // Aqui estamos retornando o JSX que representa o componente LoginPage. Ele consiste em um formulário de login com
    campos de entrada para username e password, um botão de login e uma mensagem de erro que é exibida quando ocorre
    um erro de login. As funções onChange dos campos de entrada atualizam os estados username e password conforme o
    usuário digita. O botão de login chama a função handleLogin quando é clicado.
```


REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'pages' directory containing 'dashboard.tsx'. The code editor shows the implementation of the 'DashboardPage' component, which uses 'localStorage' to manage a login state.

```
1 // pages/dashboard.tsx
2
3 import React, { useEffect } from 'react';
4 import { useRouter } from 'next/router';
5
6 const DashboardPage: React.FC = () => {
7   const router = useRouter();
8
9   useEffect(() => {
10     const isLoggedIn = localStorage.getItem('isLoggedIn');
11     if (!isLoggedIn) {
12       router.push('/login');
13     }
14   }, []);
15
16   const handleLogout = () => {
17     localStorage.removeItem('isLoggedIn');
18     router.push('/');
19   };
20
21   return (
22     <div>
23       <h1>Dashboard</h1>
24       <button onClick={handleLogout}>Logout</button>
25     </div>
26   );
27 };
28
29 export default DashboardPage;
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exemplo 01 – Analisando linha a linha o arquivo dashboard.tsx

```
1 // pages/dashboard.tsx
2
3 import React, { useEffect } from 'react';
4 import { useRouter } from "next/router";
5 // Estas linhas importam o módulo React e a função useEffect do React, bem como a função useRouter do Next.js.
6 // useEffect é um hook do React que permite realizar efeitos colaterais em componentes funcionais, enquanto useRouter
7 // é um hook do Next.js que fornece acesso ao objeto de roteamento.
8
9 const DashboardPage: React.FC = () => {
10 // Aqui estamos definindo um componente funcional chamado DashboardPage. Usamos a notação de tipo do TypeScript
11 // para indicar que este é um componente funcional React que não recebe props externas.
12
13 const router = useRouter();
14 // Aqui inicializamos o hook useRouter para acessar o objeto de roteamento do Next.js.
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exemplo 01 – Analisando linha a linha o arquivo dashboard.tsx

```
9      useEffect(() => {  
10          const isLoggedIn = localStorage.getItem('isLoggedIn');  
11          if (!isLoggedIn) {  
12              router.push('/login');  
13          }  
14      }, []);
```

// Aqui utilizamos o hook useEffect para realizar um efeito colateral. Este efeito será executado apenas uma vez, quando o componente DashboardPage for montado. Dentro do efeito, verificamos se o usuário está logado, obtendo o valor de isLoggedIn do localStorage. Se o usuário não estiver logado, redirecionamos para a página de login usando router.push('/login').

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exemplo 01 – Analisando linha a linha o arquivo dashboard.tsx

```
15
16      const handleLogout = () => {
17          localStorage.removeItem('isLoggedIn');
18          router.push('/');
19      };
20      // Aqui definimos uma função handleLogout que será chamada quando o botão de logout for clicado. Esta função
    remove a chave isLoggedIn do localStorage e redireciona o usuário de volta para a página inicial do
    aplicativo ('/').
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

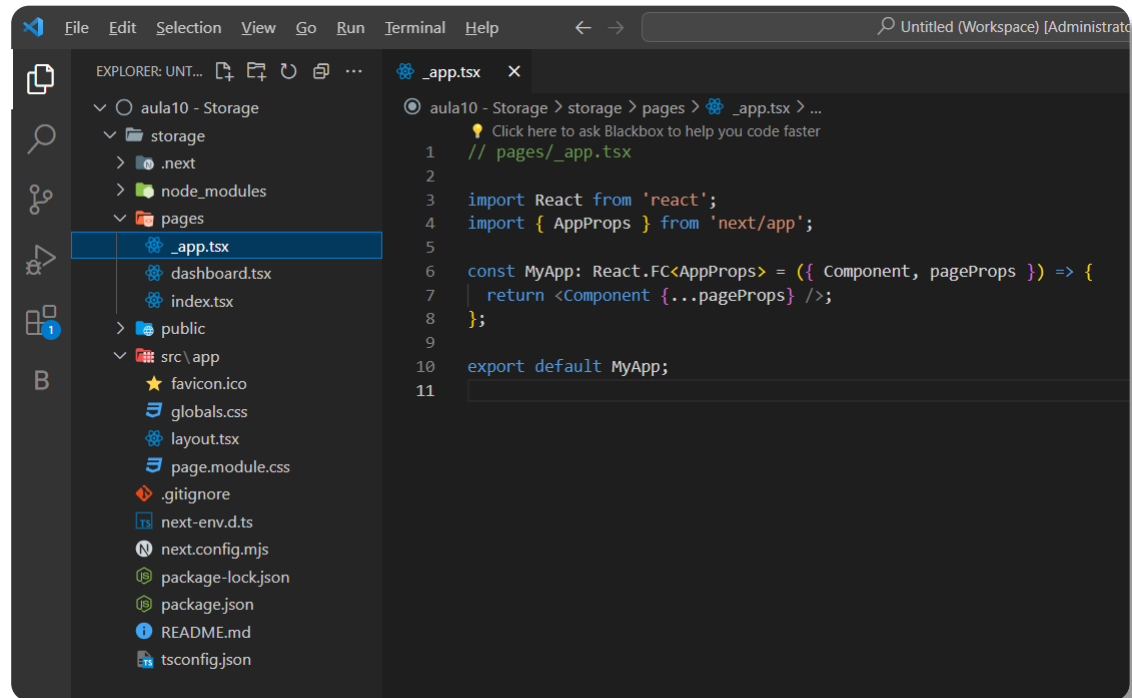
✓ Introdução ao LocalStorage e SessionStorage

✓ Exemplo 01 – Analisando linha a linha o arquivo dashboard.tsx

```
21         return (  
22             <div>  
23                 <h1>Dashboard</h1>  
24                 <button onClick={handleLogout}>Logout</button>  
25             </div>  
26         );  
27     };  
28     // Aqui estamos retornando o JSX que representa o componente DashboardPage. Ele consiste em um título h1 indicando  
29     // que é a página do Dashboard e um botão de logout que chama a função handleLogout quando é clicado.  
30  
31     export default DashboardPage;
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage



REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

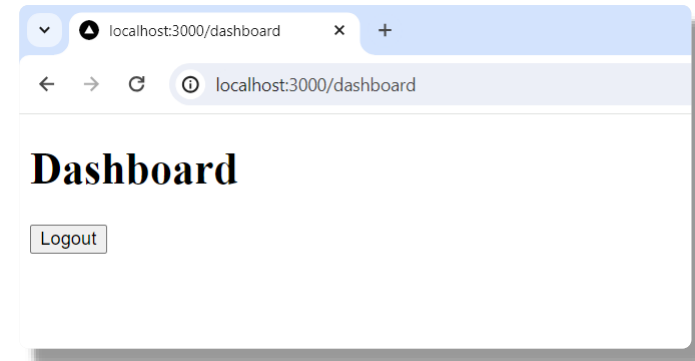
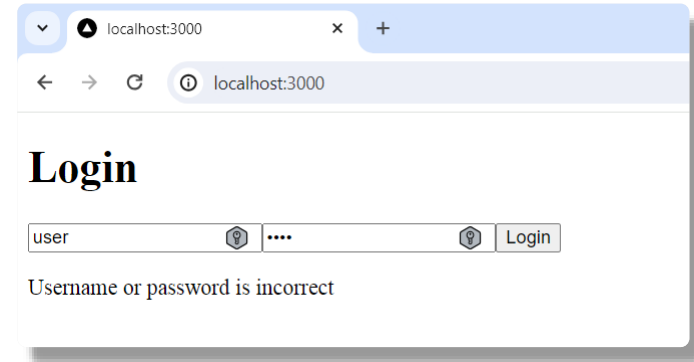
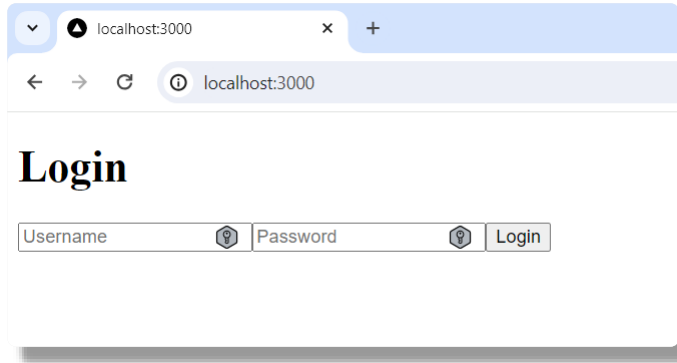
✓ Exemplo 01 – Analisando linha a linha o arquivo dashboard.tsx

```
1  // pages/_app.tsx
2
3  import React from 'react';
4
5  import { AppProps } from 'next/app';
6  // Estas linhas importam o módulo React e o tipo AppProps do Next.js. AppProps é um tipo que define as props
   padrão que são passadas para o componente _app.
7
8  const MyApp: React.FC<AppProps> = ({ Component, pageProps }) => {
9    // Aqui estamos definindo um componente funcional chamado MyApp. Usamos a notação de tipo do TypeScript para
   indicar que este é um componente funcional React que recebe props do tipo AppProps. AppProps é o tipo de props que
   são passadas para o componente _app. No componente MyApp, estamos desestruturando as props Component e pageProps
   de AppProps.
10
11    return <Component {...pageProps} />;
12    // Esta linha retorna o componente Component (que pode ser qualquer uma das páginas do seu aplicativo
   Next.js) com as props pageProps. Essencialmente, isso permite que o componente _app atue como um wrapper em
   torno de todas as outras páginas do seu aplicativo, permitindo a passagem de props comuns para todas elas.
13
14   };
15
16   export default MyApp;
```

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exemplo 01 – Resultado



REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exercício 01

- ✓ Crie uma pequena aplicação web onde o usuário possa inserir e visualizar notas. As notas devem ser armazenadas localmente no navegador utilizando o localStorage ou sessionStorage.
- ✓ **Requisitos Funcionais:**
 - ✓ Adicionar Nota: O usuário deve ser capaz de inserir uma nota através de um campo de texto e adicionar essa nota à lista de notas.
 - ✓ Visualizar Notas: Todas as notas adicionadas devem ser exibidas na tela.
 - ✓ Armazenamento Local: As notas devem ser armazenadas localmente no navegador. Isso significa que, mesmo após atualizar a página, as notas adicionadas devem persistir.
 - ✓ Remover Nota: O usuário deve ser capaz de remover uma nota da lista.

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exercício 02

- ✓ Desenvolva uma aplicação de lista de tarefas (to-do list) que permita aos usuários adicionar, marcar como concluída e remover tarefas. As tarefas devem ser armazenadas localmente no navegador usando localStorage ou sessionStorage.
- ✓ **Requisitos Funcionais:**
 - ✓ **Adicionar Tarefa:** O usuário deve poder adicionar uma nova tarefa à lista.
 - ✓ **Marcar como Concluída:** O usuário deve ser capaz de marcar uma tarefa como concluída.
 - ✓ **Remover Tarefa:** O usuário deve poder remover uma tarefa da lista.
 - ✓ **Armazenamento Local:** As tarefas devem ser armazenadas localmente no navegador. Isso significa que, mesmo após atualizar a página, as tarefas adicionadas e marcadas como concluídas devem persistir.
 - ✓ **Filtro de Tarefas:** Permita que o usuário filtre as tarefas por status (concluídas e não concluídas).

REACT – LOCALSTORAGE E SESSIONSTORAGE

✓ Introdução ao LocalStorage e SessionStorage

✓ Exercício 02 (continuação)

✓ Requisitos Técnicos:

- ✓ Utilize HTML, CSS e JavaScript para construir a aplicação.
- ✓ Utilize localStorage ou sessionStorage para armazenar as tarefas.
- ✓ Estructure sua aplicação de forma organizada, separando a lógica de apresentação da lógica de armazenamento.
- ✓ Implemente lógica para manipulação de tarefas, incluindo adição, marcação como concluída e remoção.
- ✓ Utilize recursos de manipulação do DOM para atualizar dinamicamente a lista de tarefas na interface do usuário.

REACT – LOCALSTORAGE E SESSIONSTORAGE

- ✓ **Introdução ao LocalStorage e SessionStorage**

- ✓ **Exercício 02 (continuação)**

- ✓ **Sugestões de Implementação:**

- ✓ Use um formulário para permitir que os usuários adicionem novas tarefas.
- ✓ Exiba as tarefas em uma lista, mostrando o título da tarefa e um botão para marcar como concluída.
- ✓ Ao clicar no botão para marcar como concluída, atualize o estado da tarefa e atualize a interface do usuário em conformidade.
- ✓ Forneça opções de filtro para que o usuário possa visualizar apenas tarefas concluídas, tarefas não concluídas ou todas as tarefas.
- ✓ Adicione uma opção para limpar todas as tarefas concluídas de uma vez.

REACT – LOCALSTORAGE E SESSIONSTORAGE

- ✓ **Introdução ao LocalStorage e SessionStorage**

- ✓ **Exercício 02 (continuação)**

- ✓ **Bônus (Desafio Extra):**

- ✓ Implemente a funcionalidade de edição de tarefas existentes.
- ✓ Adicione uma data de vencimento opcional para as tarefas e destaque aquelas que estão próximas da data de vencimento.
- ✓ Adicione categorias ou tags para as tarefas e permita que os usuários filtrem por categoria.
- ✓ Adicione uma opção para reordenar as tarefas manualmente.

FRONT-END DESIGN ENGINEERING

ANTONIO, C. Pro React: Build Complex Front-End Applications in a Composable Way With React. Apress, 2015.

BOSWELL, D; FOUCHER, T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. Estados Unidos: O'Reilly Media, 2012.

BRITO, Robin Cris. Android Com Android Studio - Passo A Passo. Editora Ciência Moderna.

BUNA, S. React Succinctly. Estados Unidos: [s.n], 2016. Disponível em: <www.syncfusion.com/ebooks/reactjs_succinctly>. Acesso em: 12 de janeiro de 2023.

FACEBOOK (2019a). React: Getting Started. React Docs, 2019. Disponível em: <reactjs.org/docs/react-api.html>. Acesso em: 13 de janeiro de 2023.

FACEBOOK (2019b). React Without ES6. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-es6.html>. Acesso em: 10 de janeiro de 2023.

FACEBOOK (2019c). React Without JSX. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-jsx.html>. Acesso em: 10 de janeiro de 2023.

FORM BUILDER. Build your form with code and example. Disponível em <<https://react-hook-form.com/form-builder>> . Acessado em 25 de março de 2024.

FREEMAN, Eric ROBSON, Elisabeth. Use a Cabeça! Programação em HTML5. Rio de Janeiro: Editora Alta Books, 2014

GACKENHEIMER, C. Introduction to React: Using React to Build scalable and efficient user interfaces.[s.i.]: Apress, 2015.

HUDSON, P. Hacking with React. 2016. Disponível em: <www.hackingwithreact.com/read/1/3/introduction-to-jsx>. Acesso em: 13 janeiro de 2023.

KOSTRZEWA, D. Is React.js the Best JavaScript Framework in 2018? 2018. Disponível em: <hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8>. Acesso em: janeiro de 2023.

MARTIN, R. Clean Code: A Handbook of Agile Software Craftsmanship. Estados Unidos: Prentice Hall, 2009.

MDN WEB DOCS. Guia JavaScript. Disponível em <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>>. Acessado em 29 de janeiro de 2023.

NELSON, J. Learn React's Fundamentals Without the Buzzwords? 2018. Disponível em: <jamesknelson.com/learn-react-fundamentals-sans-buzzwords>. Acesso em: 12 janeiro de 2023.

FRONT-END DESIGN ENGINEERING

NIELSEN, J. Response Times: The 3 Important Limits. 1993. Disponível em: <www.nngroup.com/articles/response-times-3-important-limits>. Acesso em: 10 janeiro de 2023.

O'REILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap>. Acesso em: 10 de janeiro de 2023.

PANDIT, N. What Is ReactJS and Why Should We Use It? 2018. Disponível em: <www.c-sharpcorner.com/article/what-and-why-reactjs>. Acesso em: 12 de janeiro de 2023.

RAUSCHMAYER, A. Speaking JavaScript: An In-Depth Guide for Programmers. Estados Unidos: O'Reilly Media, 2014.

REACTIVA. O arquivo package-lock.json. Disponível em: <<https://nodejs.reativa.dev/0020-package-lock-json/index>>. Acessado em 13 de janeiro de 2023.

_____. O guia do package.json. Disponível em: <<https://nodejs.reativa.dev/0019-package-json/index>>. Acessado em 13 de janeiro de 2023.

RICOY, L. Desmitificando React: Uma Reflexão para Iniciantes. 2018. Disponível em: <medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114>. Acesso em: 13 janeiro de 2023.

SILVA, Maurício Samy. Ajax com jQuery: requisições Ajax com a simplicidade de jQuery. São Paulo: Novatec Editora, 2009.

_____. Construindo Sites com CSS e XHTML. Sites Controlados por Folhas de Estilo em Cascata. São Paulo: Novatec, 2010.

_____. CSS3 - Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS. São Paulo: Novatec Editora, 2010.

STACKOVERFLOW. Most Popular Technologies: Web Frameworks. Developer Survey Results, StackOverflow, 2019. Disponível em: <insights.stackoverflow.com/survey/2019#technology>. Acesso em: 13 de janeiro de 2023.

TYPESCRIPT SUPPORT. Disponível em <<https://react-hook-form.com/ts#UseFormProps>> . Acessado em 25 de março de 2024.

W3C. HTML5 - A linguagem de marcação que revolucionou a web. São Paulo: Novatec Editora, 2010.

_____. A vocabulary and associated APIs for HTML and XHTML. Disponível em <<https://www.w3.org/TR/2018/SPSD-html5-20180327/>>. Acessado em 28 de abril de 2020, às 20h53min.

FRONT-END DESIGN ENGINEERING

W3C. Cascading Style Sheets, level 1. Disponível em <<https://www.w3.org/TR/2018/SPSD-CSS1-20180913/>>. Acessado em 28 de abril de 2020, às 21h58min.

_____. Cascading Style Sheets, level 2 Revision 2. Disponível em <<https://www.w3.org/TR/2016/WD-CSS22-20160412/>>. Acessado em 28 de abril de 2020, às 22h17min.

_____. Cascading Style Sheets, level 2. Disponível em <<https://www.w3.org/TR/2008/REC-CSS2-20080411/>>. Acessado em 28 de abril de 2020, às 22h03min.

_____. Cascading Style Sheets, level 3. Disponível em <<https://www.w3.org/TR/css-syntax-3/>>. Acessado em 28 de abril de 2020, às 22h18min.

_____. HTML 3.2 Reference Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html32-20180315/>>. Acessado em 28 de abril de 2020, às 19h37min.

_____. HTML 4.0 Specification. Disponível em <<https://www.w3.org/TR/1998/REC-html40-19980424/>>. Acessado em 28 de abril de 2020, às 19h53min.

_____. HTML 4.01 Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html401-20180327/>>. Acessado em 28 de abril de 2020, às 20h04min.

_____. Cascading Style Sheets, level 2 Revision 1. Disponível em <<https://www.w3.org/TR/CSS2/>>. Acessado em 28 de abril de 2020, às 22h13min.

WIKIPEDIA. JavaScript. Disponível em <<https://pt.wikipedia.org/wiki/JavaScript>>. Acessado em 29 de abril de 2020, às 10h.

YUP. Disponível em <<https://www.npmjs.com/package/yup>> . Acessado em 25 de março de 2024.