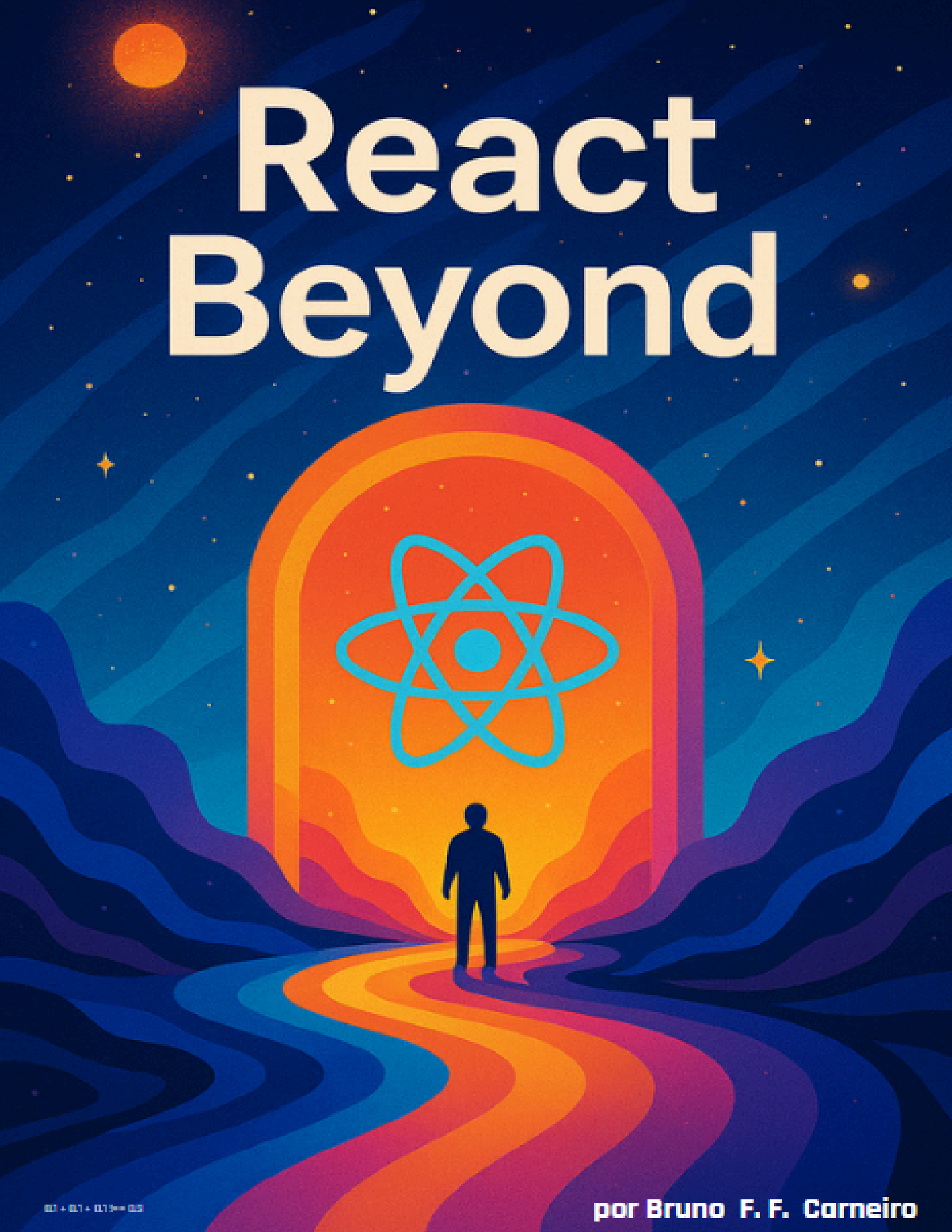
React Beyond



0.1 + 0.1 + 0.1 !== 0.3

por Bruno F. F. Corneiro

Dedicatória

Dedico este livro à minha família, que sempre acreditou em mim.
À minha esposa, que tanto amo, e que está sempre ao meu lado — mesmo nos momentos mais difíceis, mesmo quando eu mesmo duvidei.
Às minhas filhas: Cecília, com sua inteligência incrível, sua curiosidade e argumentação inabaláveis; e Marcela, que ainda está por vir, mas que eu já amo incondicionalmente. Amo vocês!
Agradeço também ao meu irmão, que sempre me apoiou em minha jornada, e aos meus pais — sem eles, eu não estaria aqui hoje, sempre cuidando de mim e me guiando para ser uma pessoa melhor, obrigado por tudo o que fizeram. Sou imensamente grato por todo carinho e amor que me deram.
Deixo ainda um agradecimento especial a algumas pessoas que me ajudaram na revisão deste livro: Kaue Drigo[1] , Carlos de Lima Junior[2] e Larissa Ferreira[3], a contribuição de vocês foi crucial para a criação deste livro, muito obrigado pela paciência e carinho com este material.

... de mim, pra mim mesmo, por Bruno Ferreira Fernandes Carneiro

[1] https://www.linkedin.com/in/kauedrigo/
[2] https://www.linkedin.com/in/carlosjunior137/
[3] https://www.linkedin.com/in/larissacostaf/

Sumário

- Sumário 2
- Como se preparar para ler este e-book 7
- JavaScript - O que é necessário 9
- Roadmap 11
- Projetos - Github 12
- Capítulo I -História 14
  - História e Evolução do React 14
  - Open Source e Crescimento 14
  - Evolução Técnica 15
  - Comunidade e Ecossistema 16
  - Reflexão 16
  - Comparação com Outras Bibliotecas/Frameworks 17
  - Adoção e Comunidade 18
  - Por que React? 20
  - Como ele funciona? 21
  - Virtual DOM 23
  - Ecossistema React 25
  - Gerenciamento de Estado 26
    - Roteamento 26
    - Estilização 26
- Capítulo II - Configurando o Ambiente de Desenvolvimento 28
  - Ferramentas e IDEs recomendadas 28
  - Versões utilizadas 29
  - Criando um projeto React 29
  - Como criar um projeto real? 31
  - Estrutura de um projeto React e explicação do boilerplate 33
  - Descrição dos Componentes Principais 36
- Capítulo III - Componentes e Props 39
  - O que é JSX? 41
    - Primeiro componente 42
  - Importação Exportação de componentes 46
    - Definição de nomes 47
    - Retorno 48
  - Passando props através dos componentes 49
  - Children e Spread 54

Uso do Spread Operator para enviar Props 56
Renderização condicional 57
Renderização de lista 61
Dumb e Smart Components 63
Entendendo Dumb Components (Componentes sem Estado): 67
Entendendo Smart Components (Componentes com Estado): 68
Outros exemplos de componentes: 70
Capítulo IV - Estado, Ciclo de Vida e Efeitos 73
O que é o Estado? 74
Atualização de Interface e Ciclo de Vida 78
Estado local 82
Ciclo de Vida 85
Fluxo de Chamadas no Chat 87
Efeitos (effects), como o React realiza o re-render e como ele é disparado: 88
Capítulo V - Gerenciamento de estado e envio de props 93
Como resolver este cenário? 96
Criação: 98
Criação de um Provider: 99
Utilização do Contexto: 99
Detalhes Importantes: 105
Cenários bastante comuns para o uso de contextApi: 106
Cuidados: 107
Gerenciadores de estado 108
Redux 109
Zustand 112
Visão crítica 113
Arquitetura Flux 115
Complexidade e Curva de Aprendizado 116
Boilerplate e Configuração 117
Debugging e Ferramentas de Desenvolvimento 118
Escalabilidade 118
Visão do autor 120
Capítulo VI - Hooks 122
Principais Hooks: 125
Outros Hooks: 125
Sendo assim, o que é um Hook? 126
State Hooks 127
Entendo o useState 127
Effect Hooks 138
Efeito sem limpeza 139
Efeitos com Limpeza 140

Exemplo prático 142
Utilizando mais efeitos 146
Custom Hooks 150
Passagem de funções para Hooks 158
Uso do hook: 159
Explicação do código: 160
Capítulo VII - Design Patterns com React 162
O que é um Design Pattern ? 162
Do que consiste um padrão de projeto? 163
Creational Design Patterns (Padrões Criacionais) 164
Structural Design Patterns (Padrões Estruturais) 164
Behavioral Design Patterns (Padrões Comportamentais) 165
Como são os Design Patterns para React? 167
O que são Higher-Order Components? 168
O que eu posso fazer com HOCs? 169
Proxy 170
Quando devo utilizar um proxy? 170
Inheritance Inversion (Herança Inversa) 173
Quando devo utilizar uma Herança Inversa? 173
Contexto: 174
Render Props 178
Vantagens: 178
Desvantagens: 179
Quando usar: 179
Quando não usar: 180
Context Api 180
Por que usar a Context API? 181
Quais são os desafios? 181
Quando usar a Context API? 182
Quando evitar? 182
Hooks 183
Capítulo VIII - Otimização e Performance 186
Performance e negócios 187
Conceitos fundamentais do React 187
Lazy Loading e Code Splitting 188
Otimização de assets 189
Uso cuidadoso de bibliotecas de terceiros 190
Memoization com React.memo 191
Qual é o formato do memo? 191
Quando usar React.memo 197
Hook useMemo 198

Hook useCallback 203
Componente filho com o memo: 204
Conclusão 208
Bundlers, Code Splitting e Lazy Loading 209
O que é Code Splitting? 209
O que é um Bundler? 212
O que é Lazy Loading? 215
Algumas formas de diminuir o tamanho desses arquivos: 217
Lazy Loading com React 217
Capítulo IX - React Router 224
O que é o React Router? 224
Mas por que rotas importam para o usuário? 226
Aplicações Web tradicionais vs SPAs 230
Client-side routing 232
Vantagens: 233
Desvantagens: 233
Server-side routing 233
Vantagens: 234
Desvantagens: 234
Aplicação do React Router na prática 235
Layouts 239
Página Dashboard 241
Aplicação Principal 242
Uso de parâmetros pela URL 243
Rota configurada com o Loader 245
Componente ProductDetails 246
Lazy Loading e Code Splitting 247
Proteção das rotas 249
Conclusão 254
Capítulo X - Tendências futuras e recursos comunitários 256
React 19.0 256
React Compiler 256
Actions 258
React Canary 260
Novo hook use() 260
Alternância entre versões: 263
Conclusão 272
Client Actions 272
Hook useOptimistic 275
Exemplo de uso do useOptimistic 276
Benefícios de useOptimistic: 278

Conclusão sobre o React 19 278
Estudos contínuos 280
Acompanhe os perfis do React Core Team 281
Explore e contribua com projetos Open Source 282
Participe de comunidades e fóruns online 282
Pratique com projetos e desafios reais 283
Crie conteúdo e compartilhe conhecimento 283
Experimente novos recursos e APIs do React 283
Erre e aprenda com os erros 284
Cuide de você 284
Domine JavaScript 285
React no mundo real 285
Conclusão 287
React 288
Front-end 288
Ferramentas 289
Conceitos Técnicos 289
Outros Termos 289

Como se preparar para ler este e-book

Antes de começar a leitura, é importante instalar algumas ferramentas para otimizar seu aprendizado. Ferramentas pré-configuradas:

- Editor de texto
- Node versão 18+
- npm, yarn ou pnpm

Procurei deixar o livro com o máximo de exemplos práticos possível. Meu objetivo aqui é mostrar como é o trabalho diário em uma empresa, aplicando as melhores práticas no desenvolvimento com React.

Para ter um maior aproveitamento, também é muito importante que o leitor tenha um bom entendimento de JavaScript (ES6 +) e algoritmo (isso mesmo, lógica de programação que hoje mal é cobrada nas empresas).

É desejável ter conhecimento de HTML 🙂 e noções básicas de CSS para facilitar o entendimento dos tópicos sobre estilização e interfaces.
É importante ir testando cada exemplo deste livro para que você absorva o conteúdo. A prática é o que vai te levar ao entendimento mais rápido dos conceitos aqui apresentados.

No capítulo II - **Configurando o Ambiente de Desenvolvimento**, apresento rapidamente como instalar o Vite[4]. O meu objetivo não é aprofundar nesta ferramenta e muito menos demonstrar como ela funciona, mas é uma alternativa para criar rapidamente o seu projeto e poder executar cada exemplo aqui apresentado.

Outra forma de testar é pela ferramenta CodeSandbox[5], muito comum na área de desenvolvimento de software. Basta escolher React para iniciar o projeto e adicionar o código aqui apresentado.

Lembre-se do princípio de Pareto 80/20: quero focar 80% na prática do dia a dia e 20% na teoria. Os fundamentos são frequentemente negligenciados, o que dificulta a aplicação prática do conhecimento adquirido.

Sem mais, que comecem os códigos!

[4] https://vite.dev/
[5] https://codesandbox.io/

JavaScript - O que é necessário

Recomendo fortemente que você tenha conhecimentos básicos de JavaScript. E quando digo conhecimentos básicos, não me refiro apenas a saber criar uma função, mas a conseguir trabalhar com JavaScript no dia a dia, utilizando recursos do ES5 [6] e ES6[7] e além.

Aqui vai uma trilha que criei sobre JavaScript: https://www.tautorn.com.br/docs/javascript/introducao.

Por que me refiro ao **ES5** e **ES6**? Porque essas versões trouxeram muitos novos recursos ao JavaScript e, na época, foram uma revolução para o desenvolvimento. Apesar da distância entre as versões, ambas introduziram muitos recursos importantes.

Portanto, tenha em mente a importância de dominar JavaScript para tirar o máximo proveito deste livro. Recomendo, inclusive, que para aprender qualquer biblioteca ou framework baseado em JavaScript, você tenha um bom domínio da linguagem. Caso contrário, você terá muita dificuldade em evoluir, e conceitos básicos da linguagem podem acabar se misturando com React e causar problemas de entendimento.

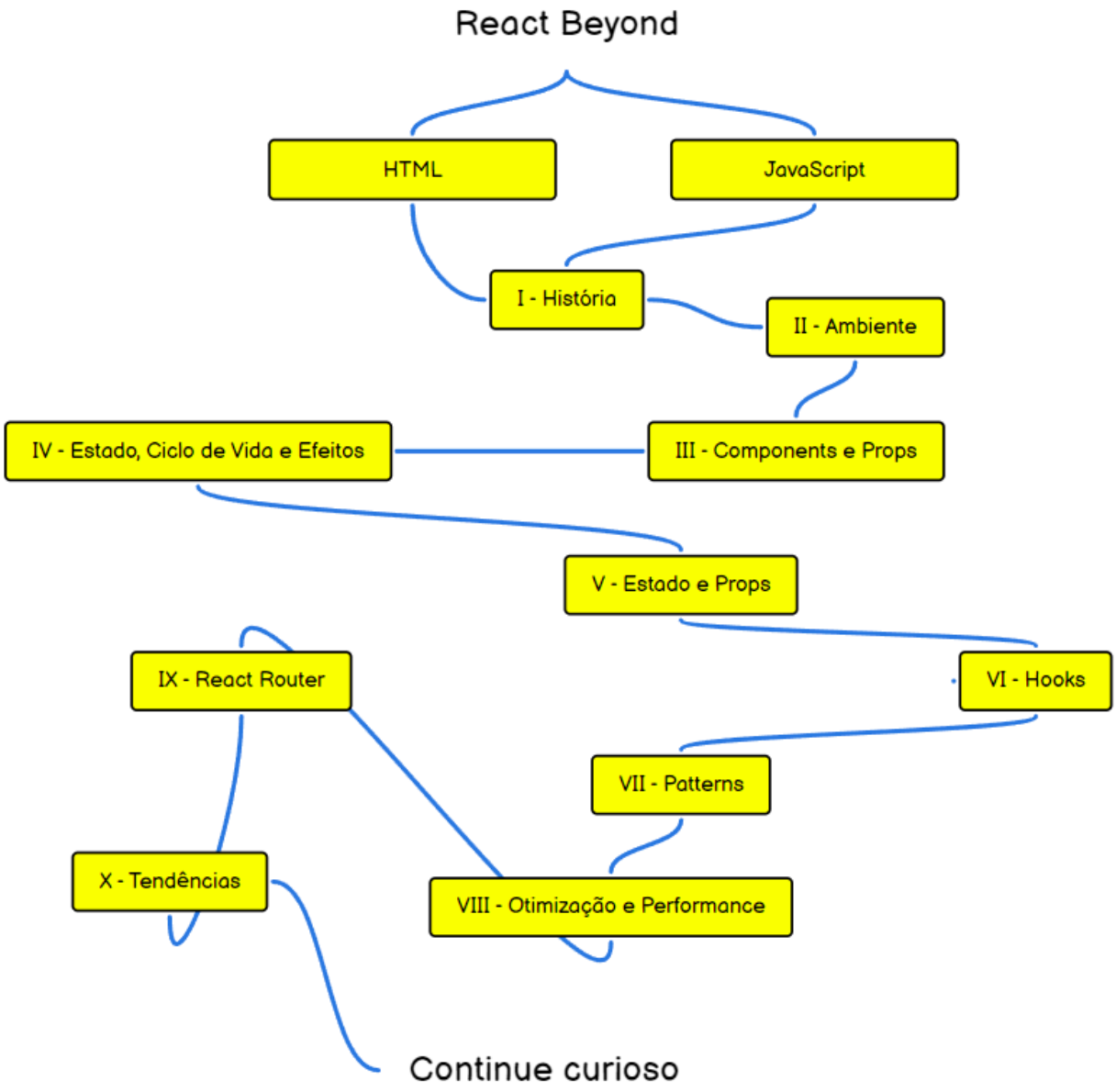
[6] https://www.tautorn.com.br/docs/javascript/versions/es2009/introducao
[7] https://www.tautorn.com.br/docs/javascript/versions/es2015/introducao

Não é necessário ser um especialista em JavaScript (acredite ou não, é comum encontrar desenvolvedores que possuem dificuldades com essa linguagem e trabalham com ReactJS e outras ferramentas no dia a dia), mas um bom entendimento e uso da linguagem são fundamentais. O mesmo vale para lógica de programação, mas não entrarei nesse ponto agora.

Quer fazer um teste de JavaScript? Acesse a ferramenta chamada Dokimi[8] que criei justamente para testar conhecimento em uma determinada ferramenta ou linguagem. Somente o teste é suficiente? Se você acertar 100% talvez, mas isso é teoria. Saiba a prática!

[8] https://dokimi.tautorn.com.br/

Roadmap



Estude bastante e divirta-se. Programar é algo mágico.

Projetos - Github

Separei no meu github exemplos práticos deste livro para facilitar o acesso e consulta: **https://github.com/Tautorn/react-beyond-ebook.git**

Obs.: É muito importante entender que desde a publicação deste livro e dos exemplos nele contidos até a data que você esteja lendo, atualizações nas bibliotecas e ferramentas utilizadas podem ocorrer e com isso estarem diferentes do que usei à época para construir este ebook.

Valorizo muito manter meus leitores sempre atualizados. No entanto, é importante entender que, após a publicação, pode ser difícil acompanhar as inúmeras mudanças e atualizações nas bibliotecas utilizadas neste projeto. Ainda assim, reforço que, no momento desta publicação, todas as ferramentas mencionadas estavam em suas versões mais recentes.

Todos os exemplos práticos estarão disponíveis no meu repositório. Por isso, é fundamental acompanhar a leitura deste eBook em conjunto com os códigos e trechos fornecidos. Nem sempre apresentarei um projeto completo, já que isso poderia aumentar a complexidade e dificultar o entendimento. Contudo, sempre que possível, adicionarei exemplos completos e totalmente funcionais.

Lembrando que é possível criar um mini ambiente para testar os exemplos em uma forma prática como descrito no começo deste livro. Mas sinta-se à vontade para testar em outras plataformas também. Recomendo muito o code sandbox[9], utilizei bastante para idealizar e testar os exemplos isoladamente.

Qualquer dúvida ou problema não deixe de entrar em contato comigo, como todo bom programador pequenos erros podem passar despercebidos, é assim que surgem bugs maravilhosos.

[9] https://codesandbox.io/dashboard/recent

Capítulo I -História

História e Evolução do React

O React foi criado por Jordan Walke[10], um engenheiro de software no Facebook. Inspirado no XHP, um framework de componentes HTML para PHP, ele desenvolveu o React como uma solução eficiente para construir interfaces de usuário dinâmicas. O React foi inicialmente utilizado no feed de notícias do Facebook em 2011 e, mais tarde, no Instagram em 2012.

Open Source e Crescimento

Em maio de 2013, o Facebook tornou o React Open Source durante a JSConf US, permitindo que a comunidade de desenvolvimento contribuísse para seu crescimento e evolução. Esta decisão marcou um ponto de virada, solidificando sua posição como uma ferramenta essencial para desenvolvedores front-end.

[10] https://x.com/jordwalke

Evolução Técnica

Desde o seu lançamento, o React passou por várias atualizações significativas, cada uma trazendo melhorias de performance, novas funcionalidades e melhores práticas de desenvolvimento. Aqui está uma breve linha do tempo da sua evolução:

- React 0.14 (outubro de 2015): Introduziu o conceito de componentes funcionais e melhorou o suporte para servidores com renderização do lado do servidor;
- React 16 (setembro de 2017): Apresentou o *Fiber*, uma reescrita completa do mecanismo de reconciliação, permitindo uma melhor gestão de prioridades, tratamento de erros e suporte para *fragments* e strings como componentes filhos;
- Hooks (fevereiro de 2019 com React 16.8): Provavelmente a mudança mais significativa recentemente, os Hooks permitiram o uso de estado e outros recursos do React em componentes funcionais, proporcionando uma forma mais simples e poderosa de construir componentes;
- React 19: Trouxe novos hooks, como o **use**, melhorias para SSR, adição de um compilador prometendo melhoras significativas de performance e memoização, além de outros recursos.

A adoção do React transformou o desenvolvimento web, promovendo a ideia de componentes reutilizáveis e declarativos. Ele inspirou a criação de várias outras bibliotecas e frameworks, como **Vue** e **Preact, Svelte** e muitas outras que adotaram e adaptaram muitos de seus conceitos iniciais. O React também estimulou o desenvolvimento de um rico ecossistema de ferramentas, bibliotecas e extensões, facilitando ainda mais a criação de aplicações complexas e interativas.

Comunidade e Ecossistema

Um dos pontos mais fortes do React é sua vibrante comunidade e ecossistema. Desenvolvedores de todo o mundo contribuem com ferramentas, bibliotecas complementares (como Redux, React Router, React Query, Styled Components, Next.js e muitas outras), e práticas recomendadas que continuam a evoluir e a moldar o futuro do React.

Reflexão

Ao refletir sobre a história e a evolução do React, é importante reconhecer como ela não apenas revolucionou o desenvolvimento web, mas também como continuamente se adapta e responde às necessidades da indústria. Este capítulo inicial não só contextualiza o React no espectro de desenvolvimento web, mas também estabelece o

palco para os leitores explorarem as capacidades mais profundas e avançadas do React nos capítulos subsequentes.

Comparação com Outras Bibliotecas/Frameworks

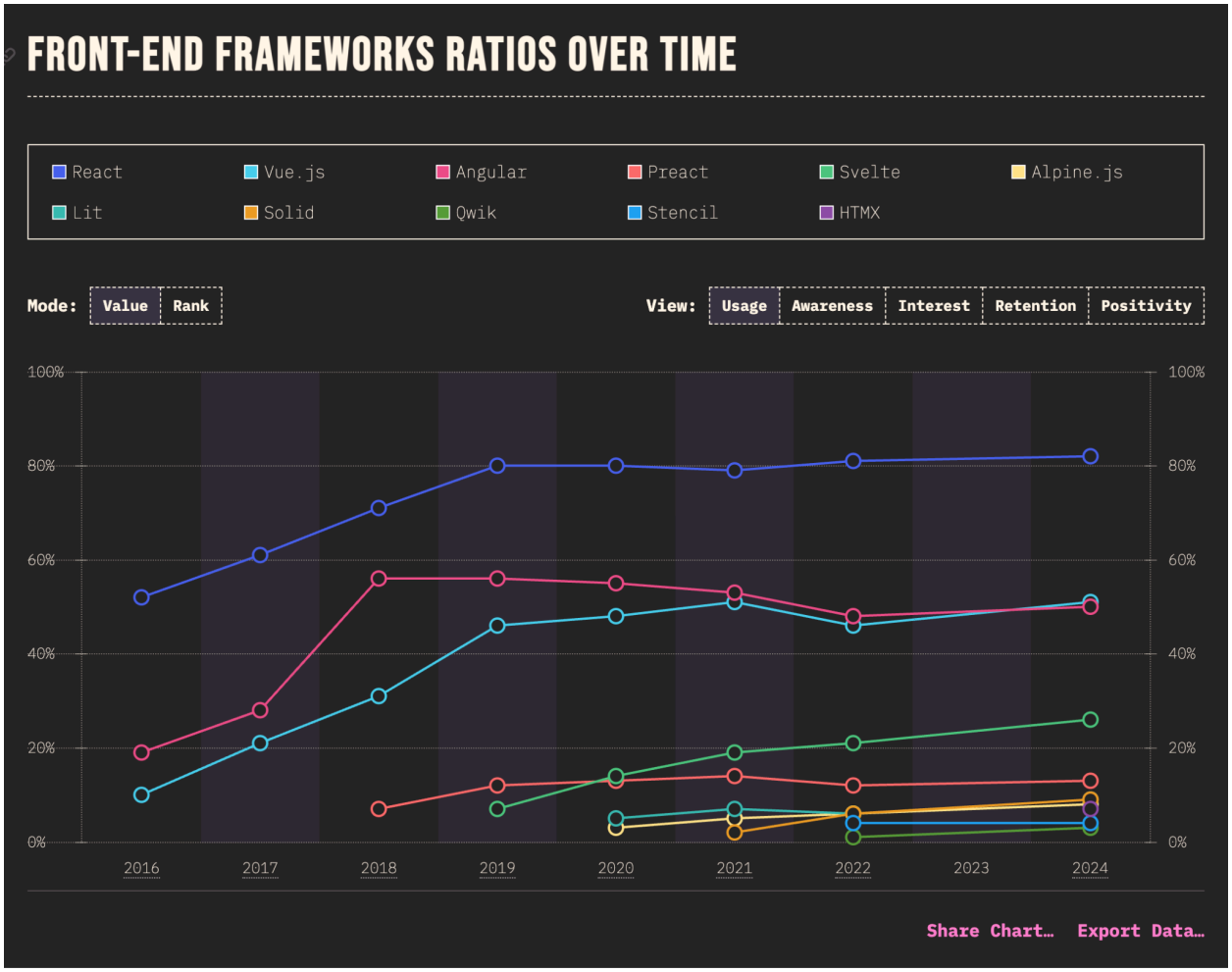
Ao comparar React com outras bibliotecas e frameworks populares, como Angular e Vue, vários fatores importantes devem ser considerados:

- **React**: Oferece alta performance com o uso do Virtual DOM, atualizações eficientes e um modelo baseado em componentes reutilizáveis;
- **Angular**: Embora seja completo e robusto, pode ser mais pesado devido à sua abordagem, o que pode impactar o desempenho em aplicações muito grandes. O framework oferece um amplo conjunto de ferramentas, APIs e bibliotecas para simplificar e otimizar seu fluxo de desenvolvimento;
- **Vue**: Equilibra leveza e eficiência, utilizando o Virtual DOM de forma semelhante ao React, mas com uma curva de aprendizado mais suave. O Vue estende o HTML padrão com uma sintaxe de template que nos permite descrever declarativamente a saída HTML com base no estado do JavaScript.

Adoção e Comunidade

- **React**: Amplamente adotado em todo o mundo, conta com uma comunidade enorme, vasto ecossistema e suporte contínuo da Meta (Facebook).
- **Angular**: Possui uma forte presença, especialmente em ambientes corporativos e aplicações de larga escala, com suporte direto do Google.
- **Vue**: Crescendo rapidamente em popularidade, é muito utilizado em projetos menores, startups e por desenvolvedores independentes, graças à sua simplicidade e flexibilidade.

Aqui vai um gráfico de uso e adoção do React ao longo do tempo comparando com algumas outras bibliotecas e frameworks voltados para o Front-end.



Fonte: https://2024.stateofjs.com/en-US/libraries/front-end-frameworks

Por que React?

Escolher React para o desenvolvimento de aplicações web traz várias vantagens:

- **Declarativo**: React facilita a criação de interfaces interativas por meio da declaração de componentes. O código torna-se mais previsível e fácil de debugar.
- **Componentes Reutilizáveis**: Facilita a construção de componentes isolados que gerenciam seu próprio estado, tornando o código mais modular e de fácil manutenção.
- **Aprendizado Direcionado**: Apesar de possuir uma curva de aprendizado inicial, uma vez compreendido, React se torna uma ferramenta poderosa com a qual é fácil trabalhar.
- **Comunidade Forte e Suporte**: Sendo uma das bibliotecas mais populares, React tem uma comunidade vibrante. Isso significa acesso a inúmeros recursos, tutoriais e suporte de outros desenvolvedores.
- **Flexível**: React pode ser usado em uma variedade de projetos, desde simples páginas Web até aplicações complexas de página única (SPA) e até mesmo em desenvolvimento móvel com React Native.
- **Performance**: Com seu virtual DOM e algoritmos inteligentes de reconciliação, React minimiza o número de manipulações de

DOM necessárias para atualizar a view, garantindo uma alta performance.

Ao escolher React, os desenvolvedores ganham acesso a um ecossistema robusto e uma comunidade ativa, enquanto aproveitam os benefícios de trabalhar com uma biblioteca flexível e performática.

Como ele funciona?

React funciona com base em um conceito chamado ***Componentização***. Isso significa que a interface do usuário pode ser dividida em pequenos blocos chamados ***componentes***. Cada componente é responsável por uma parte da interface do usuário e pode ser reutilizado em diferentes partes do aplicativo. Imagine pequenos blocos para separar a interface do usuário que podem ser reutilizados em diferentes partes do aplicativo.

A melhor comparação é com pequenas peças de lego que juntas formam algo maior.

```
const Card = () => {
  return (
    <div>
      <h1>Card</h1>
      <p>Conteúdo do card</p>
    </div>
  )
}

const Button = () => {
  return <button>Botão</button>;
}

const App = () => {
  return (
    <div>
      <Card />
      <Button />
    </div>
  )
}
```

É claro que isso é uma simplificação, mas é fundamental o entendimento desses conceitos para que você possa começar a trabalhar com React.

Todavia, para ter um entendimento claro de como funciona o core do React precisamos entender sobre **Virtual DOM**, **Reconciliação** e

Algoritmo de diferenciação. Esses são conceitos fundamentais para entender profundamente como o React funciona e como ele consegue ser tão performático e escalável.

Além disso, esses conceitos podem fazer você ter uma clareza muito maior de como desenvolver as aplicações, ter ganhos de performance, lógica, separação de código, escala de arquitetura e outros benefícios.

Virtual DOM

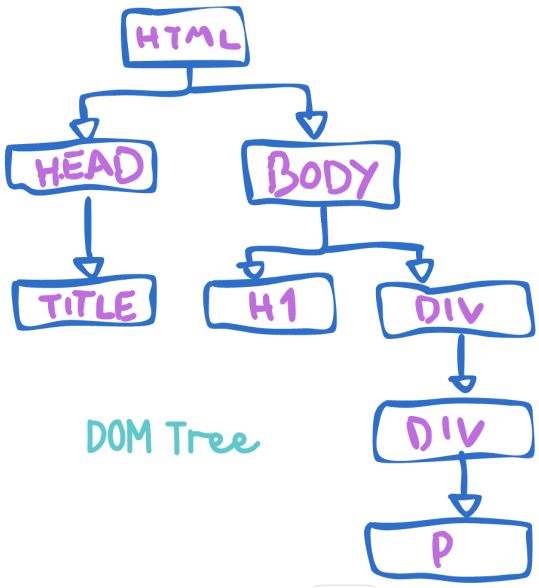
Para quem trabalha com desenvolvimento web deve estar acostumado com o *DOM* (Document Object Model). O DOM é uma interface de programação para documentos HTML e XML. Ele representa a página para os navegadores poderem alterar a estrutura, estilo e conteúdo da página. Algo como:

```
<!doctype html>
<html>
  <head>
    <title>Minha página</title>
  </head>
  <body>
    <h1>Olá mundo</h1>
    <div>
      <div>
        <p>Conteúdo</p>
```

```
      </div>
    </div>
  </body>
</html>
```

O DOM é uma árvore de elementos que representa a estrutura da página.

Cada elemento é um nó na árvore e pode ser acessado e manipulado por meio de JavaScript.



O problema é que manipulações diretamente no DOM são bastante onerosas para a aplicação, difíceis de controlar e podem causar efeitos colaterais. Mas o principal ponto aqui é a lentidão que isso pode causar.

O Virtual DOM é uma abstração do DOM. Ele é uma cópia da árvore de elementos que representa a estrutura da página. Quando o estado de um componente muda, o React cria uma nova árvore de elementos e a compara com a árvore anterior. Ele então identifica as diferenças

entre as duas árvores e atualiza apenas as partes que mudaram no DOM real.

Toda essa parte inicial vai ser melhor abordada nos próximos capítulos.

Ecossistema React

O ecossistema do React é vasto e está continuamente evoluindo, oferecendo uma ampla gama de ferramentas, bibliotecas e extensões que facilitam o desenvolvimento de aplicações.

- **Vite**: Uma solução moderna e rápida para configurar ambientes de desenvolvimento React, oferecendo carregamento ultrarrápido e suporte nativo a ESModules;
- **Create React App (CRA)**: A ferramenta oficial para criar novas aplicações React com zero configuração inicial. Embora seja funcional, **não** é mais recomendada para novos projetos devido a alternativas mais modernas;
- **Next.js**: Um framework de produção baseado em React que facilita a criação de aplicações web otimizadas, com suporte integrado para renderização no lado do servidor (SSR) e geração de sites estáticos (SSG);
- **Gatsby**: Um gerador de sites estáticos baseado em React, ideal para criar websites e aplicações extremamente rápidas e

otimizadas para SEO.

Gerenciamento de Estado

- **Zustand** e **Recoil**: Alternativas mais leves e modernas para gerenciamento de estado, que se integram perfeitamente ao React;
- **Redux**: Uma biblioteca popular para gerenciamento de estado previsível, frequentemente usada com React, especialmente em aplicações de grande escala.
- **TanStack Query:** Uma biblioteca para gerenciamento de dados e cache, facilitando a busca, sincronização e armazenamento de dados remotos, oferecendo uma abordagem eficiente para lidar com requisições HTTP e seu estado.

Roteamento

- **React Router**: A solução padrão para gerenciamento de rotas em aplicações React, permitindo a criação de rotas dinâmicas e complexas de forma simples e declarativa;

Estilização

- **Styled-components** e **Emotion**: Bibliotecas CSS-in-JS amplamente utilizadas, que oferecem uma maneira eficiente e flexível de estilizar componentes React;
- **Material-UI (MUI)**, **Ant Design** e **Chakra UI**: Bibliotecas

completas de componentes de interface, que fornecem elementos estilizados prontos para uso, facilitando o desenvolvimento de interfaces modernas;

- **Shadcn**: Uma coleção de componentes modernos e acessíveis, com integração perfeita ao Tailwind CSS, ideal para a criação de interfaces elegantes e altamente personalizáveis.

Este ecossistema rico não apenas simplifica o desenvolvimento de aplicações robustas e escaláveis, mas também fornece flexibilidade e opções para desenvolvedores de todos os níveis.

Existem milhares de outras bibliotecas, mas as listadas acima são muito utilizadas. Enquando escrevo este e-book outras 500 bibliotecas JavaScript estão sendo criadas.

Antes de sair instalando em seu projeto, valide a real necessidade. É muito comum adicionar várias dependências sem a real necessidade de uso.

Comece com o projeto o mais simples possível e há medida da **NECESSIDADE** adicione mais.

Capítulo II - Configurando o Ambiente de Desenvolvimento

Capítulo III – Componentes e Props

Capítulo IV – Estado, Ciclo de Vida e Efeitos

Capítulo V - Gerenciamento de estado e envio de props

Capítulo VI – Hooks

Capítulo VII - Design Patterns com React

Capítulo VIII – Otimização e Performance

Capítulo IX – React Router

Capítulo X - Tendências futuras e recursos comunitários

Conclusão

Dicionário de palavras

React

- **ReactJS:** Biblioteca JavaScript para construção de interfaces de usuário.
- **Componentes:** Blocos reutilizáveis que compõem a interface do usuário.
- **Hooks:** APIs que permitem gerenciar estado e efeitos colaterais em componentes funcionais.
- **Context API:** Sistema de gerenciamento de estado global no React.
- **Props:** Propriedades passadas para componentes React para comunicação entre eles.
- **State:** Estado interno de um componente React que pode ser alterado durante a execução.
- **useState:** Hook para manipulação de estado em componentes funcionais.
- **useEffect:** Hook que permite a execução de efeitos colaterais em componentes funcionais.
- **useContext:** Hook para acesso ao contexto do React.
- **React Router:** Biblioteca para gerenciamento de rotas em aplicações React.
- **Lazy Loading:** Técnica para carregar componentes sob demanda, melhorando a performance.

Front-end

- **JavaScript:** Linguagem de programação usada para desenvolvimento web.
- **TypeScript:** Superset do JavaScript que adiciona tipagem estática.
- **HTML:** Linguagem de marcação para estruturar páginas web.
- **CSS:** Linguagem de estilos para personalizar a aparência de páginas web.

- **SPA (Single Page Application):** Aplicação web onde a navegação ocorre sem recarregar a página.
- **SSR (Server Side Rendering):** Renderização do conteúdo no servidor antes de enviá-lo ao navegador.

Ferramentas

- **Node.js:** Ambiente de execução JavaScript do lado do servidor.
- **Vite:** Ferramenta para criação e desenvolvimento rápido de projetos front-end.
- **Jest:** Framework de testes para aplicações JavaScript.
- **Vitest:** Ferramenta de testes moderna otimizada para Vite e TypeScript.

Conceitos Técnicos

- **Virtual DOM:** Representação virtual do DOM usada pelo React para otimizar atualizações de interface.
- **Memoização:** Técnica para otimizar cálculos armazenando resultados já computados.
- **Re-renders:** Processo de atualização da interface do usuário pelo React.
- **Hydration:** Processo de reaproveitamento do HTML renderizado no servidor pelo React no lado do cliente.

Outros Termos

- **Comunidade:** Grupos de desenvolvedores que compartilham conhecimento e experiências.
- **Open Source:** Software de código aberto que pode ser utilizado e modificado livremente.
- **Networking:** Construção de relações profissionais na área de tecnologia.

- **Síndrome do Impostor:** Sensação de não ser bom o suficiente, mesmo tendo habilidades e conhecimento na área.