Guia Ágil

TechGuide - Alura

React

Nivel 1

- [ ] **HTML - Fundamentos**:

- HTML é uma linguagem de marcação que define a estrutura do seu conteúdo. HTML consiste em uma série de elementos que você usa para mostrar algo de uma determinada maneira ou agir de uma certo modo. As tags podem criar um hiperlink de uma palavra ou imagem para outro lugar, podem colocar palavras em itálico, podem aumentar ou diminuir a fonte e assim por diante.
- Aprender quais tags são necessárias para um HTML básico
- Criar um parágrafo de texto
- Exibir uma imagem
- Conhecer a diferença entre 'h1', 'h2', 'h3', etc
- Criar um texto com hyperlink
- Criar um formulário com campos relevantes
- Criar uma lista de itens ordenada ou não ordenada
- Criar uma lista de itens dentro de uma lista suspensa (dropdown list)
- Conectar com um arquivo de CSS
- Criar uma tabela
- Adicionar IDs e classes

- [ ] **CSS - Fundamentos**:
    - Cascading Style Sheets (CSS) é uma linguagem usada para descrever a apresentação de um documento escrito em uma linguagem de marcação como HTML ou XML. CSS pode ser usado para estilos de texto de documentos muito básicos — por exemplo, para alterar a cor e o tamanho de títulos e links. Ele pode ser usado para criar um layout — por exemplo, transformar uma única coluna de texto em um layout com uma área de conteúdo principal e uma barra lateral para informações relacionadas. Pode até ser usado para efeitos como animações.
    - Aprender a estrutura visual de uma página, com 'margin' e 'padding'
    - Estabelecer o tamanho com 'width' e 'height'
    - Aprender sobre a posição de um elemento ('static', 'relative' ou 'absolute)
    - Aprender sobre o 'display' de exibição de um elemento ('block', 'inline', 'inline-block')
    - Aprender a posicionar imagens em relação ao texto
    - Aprender sobre alinhamento
    - Aprender sobre estilo de fontes
    - Aprender as diferenças e vantagens de usar as diferentes unidades de medida em CSS (%, relativas, etc)
    - Conectar com os elementos (IDs, classes) de um arquivo HTML
    - Alterar características de um elemento quando o mouse passar por cima dele ('hover')
    - Aprender box-sizing
    - Aprender Flexbox
    - Aprender Grid
- [ ] **JavaScript - Fundamentos**:
    - JavaScript é a linguagem de programação mais popular do mundo e é uma das principais tecnologias da World Wide Web, juntamente com HTML e CSS. Ela possui tipagem dinâmica, orientação a objetos baseada em

protótipos e funções de primeira classe. Ela é multi-paradigma e suporta estilos de programação orientados a eventos, funcionais e imperativos.

- Conhecer os tipos primitivos
- Declarar variáveis, considerando a diferença entre 'var', 'let' e 'const'
- Usar estruturas condicionais ('if', 'else')
- Conhecer os operadores de atribuição e comparação ('=', '==', '===')
- Usar estruturas de repetição e laços ('while', 'for')
- Usar funções, passando parâmetros e argumentos
- Manipular arrays e listas
- Aprender o conceito de Orientação a Objetos
- Fazer um CRUD
- Obter dados de uma API
- Fazer chamadas assíncronas usando 'Async/Await', 'Promise', etc

- [ ] **DOM - Fundamentos**:

- O Document Object Model (DOM) é uma interface de programação para documentos web. Ele representa a página para que os programas possam alterar a estrutura, o estilo e o conteúdo do documento. O DOM representa o documento como nós e objetos; dessa forma, linguagens de programação podem interagir com a página.
- Entender como funciona a árvore do DOM
- Acessar e manipular elementos do HTML e CSS
- Acessar os pais e filhos de um elemento
- Inserir um novo elemento na árvore
- Remover um elemento da árvore
- Esperar por um evento em certo elemento da página usando 'addEventListener()'

- [ ] **Conceitos SPA**:

- Um aplicativo de página única (SPA - single-page application) é uma aplicação web ou site que interage com o usuário reescrevendo dinamicamente a página web atual com novos dados do servidor web, em vez do método padrão de um navegador que carrega novas páginas inteiras. O objetivo são transições mais rápidas, que tornam o site mais parecido com uma aplicação nativa.
- Entender o que é uma SPA
- Estabelecer rotas para outras páginas
- Conhecer frameworks SPA
- Comunicação com APIs

- [ ] **React - Componentes**:

- O React permite definir componentes como classes ou funções. Componentes definidos como classes fornecem mais recursos. Eles aceitam entradas arbitrárias (chamadas "props") e retornam elementos React descrevendo o que deve aparecer na tela.
- Para que servem e como funcionam componentes
- Conhecer a biblioteca Styled Components

- [ ] **React - Props**:

- Props são um objeto que é injetado em componentes e fornece alguns dados que podem ser compartilhados entre outros componentes em um fluxo de dados unidirecional, de um elemento pai para um elemento filho. Props são de somente leitura.
- Como passar props
- Como manipular props

- [ ] **React Hooks - State**:

- O hook React `useState` nos permite rastrear o estado em um componente de função. O hook `useState` pode ser usado para acompanhar strings, números, booleanos, arrays, objetos, etc.
- Controlar o estado de componentes

- Manipular variáveis
- Atualizar o valor de elementos

- [ ] **Criando uma aplicação React**:

- Estruturar um novo projeto React
- Criar uma aplicação funcional do zero

- [ ] **React Hooks - Effect**:

- UseEffect() é um hook React que permite que você manipule efeitos colaterais em seus componentes funcionais do React.
- Executar um componente somente após a renderização
- Acessar as props de um elemento
- Fazer chamadas a APIs

Nivel 2

- [ ] **React Hooks - Memo**:

- O hook React `useMemo` retorna um valor memoizado. `useMemo` apenas recalculará o valor memoizado quando uma das dependências for alterada.
- Controlar o estado de variáveis externas
- Evitar cálculos caros em cada renderização

- [ ] **React Hooks - Callback**:

- O hook React `useCallback` retorna uma função de callback memoizada.
- Retornar uma versão memoizada do callback caso as entradas tenham sido alteradas

- [ ] **React Hooks - Ref**:

- O hook `useRef` permite que você persista valores entre renderizações.
- Criar objetos mutáveis

- [ ] **React - Bibliotecas de Design System**:

- Criar e manter bibliotecas que serão consumidas e usadas como padrão para a construção de um projeto

- [ ] **React Developer Tools**:

- React Developer Tools é usado para inspecionar componentes do React, editar props e estado, e também para identificar problemas de desempenho.
- Executar o 'debugging' de aplicações

- [ ] **Versionamento Semântico para Front-end**:

- SemVer (Versionamento Semântico) é um conjunto simples de regras e requisitos que ditam como os números de versão são atribuídos e incrementados.
- Organizar as dependências de um projeto

- [ ] **CSS-in-JS**:

- CSS-in-JS refere-se a um padrão onde o CSS é composto usando JavaScript em vez de ser definido em arquivos externos.
- Lidar com código CSS usando JavaScript
- Utilizar styled-components
- Conhecer Styled-JSX

- [ ] **Styled Components**:

- Lidar com código CSS em componentes

- [ ] **React - Roteamento web**:

- Manipular a navegação entre interfaces e componentes

- [ ] **React - Gerenciamento de Estados**:

- O gerenciamento de estado em React é a prática de controlar e atualizar os dados de uma aplicação React para garantir que os componentes tenham acesso às informações necessárias. Isso é essencial para criar aplicativos interativos e dinâmicos.

- Garante que os dados exibidos nos componentes estejam sempre atualizados e em sincronia.
- Permite que componentes compartilhem dados e comuniquem-se uns com os outros.
- Otimiza a renderização, garantindo que apenas as partes relevantes da interface do usuário sejam atualizadas.
- Torna o código mais organizado e fácil de entender, especialmente em aplicativos complexos.

☐ **TypeScript - Fundamentos**:

- TypeScript é uma linguagem de programação fortemente tipada que se baseia em JavaScript.
- Entender a fundo o que são tipos e a importância da tipagem
- Aprender o que é o TypeScript, por que foi criado, como ele funciona e sua relação com o JavaScript
- Conhecer as ferramentas do TypeScript (integração com o editor de código, verificador estático e compilador)
- Escrever código em TypeScript com suas ferramentas (interfaces, enum, decorators, etc)
- Desenvolver aplicações em TypeScript

☐ **React Testing Library**:

- A React Testing Library se baseia na DOM Testing Library adicionando APIs para trabalhar com componentes React. É um conjunto de auxiliares que permitem testar componentes React sem depender de seus detalhes de implementação.
- Testar componentes React

☐ **React - Jest**:

- Jest é um framework de teste em JavaScript projetado para garantir a correção de qualquer código JavaScript. Ele permite que você escreva

testes com uma API acessível, familiar e rica em recursos que lhe dá resultados rapidamente.

- Testar componentes

- [ ] **Cypress**:

- Criar e executar testes

- [ ] **JavaScript - Callbacks e Promises**:

- Uma promessa (Promise) é um proxy para um valor não necessariamente conhecido quando a promessa é criada. Isso permite que métodos assíncronos retornem valores como métodos síncronos - em vez de retornar imediatamente o valor final, o método assíncrono retorna uma promessa de fornecer o valor em algum momento no futuro.
- Uma função de Callback é uma função passada para outra função como um argumento, que é então invocado dentro da função externa para completar algum tipo de rotina ou ação.
- Uma função assíncrona (async) é uma função declarada com a palavra-chave `async`, e a palavra-chave `await` é permitida dentro dela. As palavras-chave `async` e `await` permitem que o comportamento assíncrono e baseado em promessas seja escrito em um estilo mais limpo, evitando a necessidade de configurar explicitamente as cadeias de promessas.
- Entender o conceito de assincronicidade em programação
- Escrever código assíncrono entendendo o conceito de promessas em JavaScript
- Utilizar os métodos, palavras-chaves e objetos do JavaScript para manipulação de promessas como 'Async/Await', '.then()', 'Promise', etc
- Aprender em quais situações é necessário o uso de programação assíncrona
- Fazer chamadas em APIs com `fetch()`

- [ ] **JavaScript - Manipulação de Erros**:

- O tratamento de erros refere-se aos procedimentos de resposta e recuperação de condições de erro presentes em um aplicativo de software.

Em outras palavras, é o processo composto de antecipação, detecção e resolução de erros de aplicação, de programação ou de comunicação.

- Conhecer e tratar as exceções mais comuns
- Saber quais os tipos de erros e em quais situações eles podem ocorrer
- Entender como o Node.js faz o manejo de erros
- Usar 'try' e 'catch' para tratamento de erros
- Em que ocasiões e de que forma utilizar o `throw`
- Criar exceções específicas de acordo com a necessidade de sua aplicação

- [ ] **Babel - Fundamentos**:

- Babel é um compilador JavaScript. É uma ferramentas usada principalmente para converter código ECMAScript 2015+ em uma versão compatível com versões anteriores do JavaScript em navegadores ou ambientes atuais e mais antigos.
- Compreender como Babel converte a sintaxe para JavaScript

Nivel 3

- [ ] **Lottie**:

- Lottie é uma biblioteca que processa animações do Adobe After Effects exportadas como json com Bodymovin e as renderiza nativamente em dispositivos mobile e na web.

- [ ] **Framer Motion**:

- Criar animações

- [ ] **Service Workers**:

- Melhorar a performance e trabalhar offline;
- Permitem que aplicações web armazenem recursos no cache;
- Interceptam e gerenciam requisições de rede;
- Facilitam notificações push, mesmo quando o app não está aberto ou usuário não está na página;

- Funcionam como um intermediário entre o navegador e a rede/servidor;
- São assíncronos e não acessam diretamente o DOM, garantindo segurança e eficiência;

- [ ] **React - Validação de formulários**:

- Em React, é super importante que, em formulários, os dados inseridos pelos usuários atendam a critérios específicos antes do envio ou processamento das informações.
- Conhecer o Formik e o React Hook Form, bibliotecas populares para lidar com formulários em React.
- Conhecer o Yup e o Zod, ambas bibliotecas de validação baseadas em esquema. Permitem a definição de esquemas para validar valores de um objeto.
- Compreender como as bibliotecas trabalham, facilitando a criação de formulários complexos com validação e gerenciamento de estado.

- [ ] **React - Gerenciamento de Performance**:

- Em React, uma boa performance garante uma experiência de usuário fluida e responsiva.
- A otimização de performance envolve técnicas para minimizar a quantidade de renderizações e maximizar a eficiência da atualização da interface.
- Entender o que é otimização de performance em React.
- Conhecer e saber usar técnicas como Lazy Loading e Code Splitting.
- Conhecer as demais ferramentas e práticas para otimização de performance no React.

- [ ] **Lodash**:

- Lodash é uma biblioteca de utilitários JavaScript moderna que oferece modularidade, desempenho e extras. Os métodos modulares do Lodash são ótimos para iterar arrays, objetos e strings; manipular e testar valores; e criar funções compostas.

- [ ] **GraphQL**:

- GraphQL é uma linguagem de consulta e manipulação de dados de código aberto para APIs. É considerada uma alternativa para arquiteturas REST.
- Aprender o que é GraphQL e por que foi criado
- Entender como o GraphQL é utilizado no desenvolvimento de APIs
- Criar APIs utilizando as bibliotecas e frameworks para GraphQL

☐ **Apollo Client**:

- Apollo Client é uma biblioteca abrangente de gerenciamento de estado para JavaScript que permite gerenciar dados locais e remotos com o GraphQL.
- Utilizar o Apollo para criar um servidor GraphQL
- Conectar com uma API

☐ **React - Arquitetura**:

- Manter a arquitetura do nosso projeto bem estruturada, nos ajuda a organizar o código da melhor maneira possível para facilitar a manutenção e escalabilidade. Combinada aos princípios do SOLID, ela potencializa ainda mais toda essa organização minimizando o acoplamento e aprimorando a coesão dos componentes da nossa aplicação.
- Estruturar o projeto React com Arquitetura Limpa;
- Conhecer e aplicar os princípios SOLID para maior reusabilidade e coesão;
- Isolar responsabilidades com hooks e injeção de dependências.

☐ **Redux Saga**:

- Redux Saga é uma biblioteca que visa tornar os efeitos colaterais da aplicação (ou seja, coisas assíncronas, como busca de dados e coisas impuras, como acessar o cache do navegador) mais fáceis de gerenciar, mais eficientes de executar, fáceis de testar e melhores no tratamento de falhas.
- Criar e manter chamadas assíncronas e efeitos

☐ **NextJS - Fundamentos**:

- Construir interfaces Web

- Diminuir o tempo de carregamento das páginas
- Renderizar páginas no lado do servidor
- Melhorar a performance em React
- Construir rotas de API com funções serveless
- CSS-in-JS

Habilidade Auxiliar: Infraestrutura e Back-end

- [ ] **Git e GitHub - Fundamentos**:

- Git é um sistema de controle de versão distribuído gratuito e de código aberto projetado para lidar com tudo, desde projetos pequenos a muito grandes com velocidade e eficiência.
- GitHub é um serviço de hospedagem para desenvolvimento de software e controle de versão usando Git.
- Criar um repositório
- Clonar um repositório
- Fazer commit, push e pull de e para o repositório
- Reverter um commit
- Criar branches e pul requests
- Lidar com merge e conflitos

- [ ] **HTTP - Fundamentos**:

- HTTP significa Hyper Text Transfer Protocol. A comunicação entre computadores cliente e servidores web é feita enviando solicitações HTTP e recebendo respostas HTTP.
- Entender a diferença dos verbos HTTP
- Testar os requests e ver os status codes no navegador
- Saber fazer uma requisição HTTP na linha de comando com WGET
- Baixar uma imagem com WGET
- Fazer um post

- [ ] **JSON**:
  - JSON significa JavaScript Object Notation (notação de objeto JavaScript). É um formato de texto para armazenar e transmitir dados.
  - Criar um objeto
  - Transformar um objeto em uma string
  - Transformar uma string em objeto
  - Manipular um objeto
- [ ] **Linha de comando - Fundamentos**:
  - CLI é um programa de linha de comando que aceita entradas de texto para executar funções do sistema operacional.
  - Conhecer os principais comandos
- [ ] **Cloud - Fundamentos**:
  - Cloud, ou computação em nuvem é a distribuição de serviços de computação pela Internet usando um modelo de preço pago conforme o uso. Uma nuvem é composta de vários recursos de computação, que abrangem desde os próprios computadores (ou instâncias, na terminologia de nuvem) até redes, armazenamento, bancos de dados e o que estiver em torno deles. Ou seja, tudo o que normalmente é necessário para montar o equivalente a uma sala de servidores, ou mesmo um data center completo, estará pronto para ser utilizado, configurado e executado.
  - Conhecer a diferença entre IaaS, PaaS e SaaS
  - Conhecer os maiores provedores de cloud
  - Especializar-se em algum provedor
- [ ] **YARN**:
  - Yarn é um gerenciador de pacotes para seu código. Ele permite que você use e compartilhe código com outros desenvolvedores. O código é compartilhado por meio de algo chamado pacote (às vezes chamado de módulo). Um pacote contém todo o código que está sendo compartilhado, bem como um arquivo package.json que descreve o pacote.

- Gerenciar pacotes
- Gerenciar dependências
- Instalação de pacotes offline
- Comandos
- Arquivo yarn.lock

Habilidade Auxiliar: UX e Design

- [ ] **Design System**:

- Um Design System (sistema de design) é uma coleção de componentes reutilizáveis, guiados por padrões claros, que podem ser colocados juntos para construir aplicações.
- Criar e manter bibliotecas que serão consumidas e usadas como padrão para a construção de um projeto
- Design tokens
- Estilos fundamentais
- Construção de componentes
- Microinterações
- Documentação

- [ ] **Figma - Fundamentos**:

- Figma é uma aplicação web colaborativa para design de interfaces. O conjunto de recursos do Figma se concentra na interface do usuário e no design da experiência do usuário, com ênfase na colaboração em tempo real, utilizando uma variedade de editores de gráficos vetoriais e ferramentas de prototipagem.
- Criar layouts de páginas e componentes

- [ ] **Componentes de design**:

- Conhecer os componentes descrevem um layout ou interface

- [ ] **Sistemas de cores**:

- Definir uma paleta de cores que faça sentido para determinada interface

- [ ] **Como usar fontes**:

- Escolher a fonte mais adequada para determinado projeto
