Guia Ágil

TechGuide - Alura

React Native

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 Native - Fundamentos**:

- React Native é um framework de desenvolvimento de software de código aberto criado pelo Facebook. Ele permite desenvolver aplicativos para Android e iOS usando um único código-fonte.
- No React Native, as interfaces são construídas usando componentes. Esses componentes, como View, Text, Image, Button e TextInput, são responsáveis por definir tanto a aparência quanto o comportamento da sua aplicação. A ideia é simplesmente descrever como a interface deve se comportar de acordo com o estado da aplicação.

- [ ] **React Native - Pacotes Essenciais**:

- O React Native permite o uso de pacotes e bibliotecas de terceiros para agilizar o desenvolvimento e adicionar funcionalidades extras aos aplicativos.
- Conhecer e saber utilizar o NPM (Node Package Manager) para gerenciar pacotes.

- Conhecer os principais pacotes do React Native e suas utilidades.

- [ ] **React Native - Testes**:

- Aprender as melhores práticas de testes em React Native.
- Conhecer as ferramentas e bibliotecas de testes para React Native, como Jest e React Native Testing Library.
- Aprender a escrever testes unitários, de integração e de interface de usuário para garantir a qualidade do código.

Nivel 3

- [ ] **React Native - Nativo**:

- Conhecer as especificidades do desenvolvimento nativo para Android.
- Conhecer as especificidades do desenvolvimento nativo para iOS

- [ ] **React Native - Animações**:

- Entender os conceitos básicos de animação no React Native.
- Aprender a utilizar as bibliotecas de animação mais comuns no React Native.
- Integrar animações em componentes e fluxos de navegação.

- [ ] **React Native - Arquitetura**:

- Compreender as melhores práticas de arquitetura para criar aplicativos React Native.
- Conhecer os principais padrões de arquitetura no React Native.
- Aprender como organizar pastas e componentes.

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

- [ ] **Design Patterns**:

- Na engenharia de software, um "padrão de projeto" (Design Pattern em inglês) é uma solução geral e reutilizável para um problema que ocorre normalmente dentro de um determinado contexto de projeto de software.

- Conhecer e aplicar os principais Design Patterns

- [ ] **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

- [ ] **SOLID**:

- O Solid possui cinco princípios considerados como boas práticas no desenvolvimento de software que ajudam os programadores a escrever os códigos mais limpos, separando as responsabilidades, diminuindo acoplamentos, facilitando na refatoração e estimulando o reaproveitamento do código.

- [ ] **Clean Architecture**:

- A Clean Architecture (Arquitetura Limpa) é uma forma de desenvolver software, de tal forma que apenas olhando para o código fonte de um programa, você deve ser capaz de dizer o que o programa faz.

- [ ] **Firebase**:

- O Firebase é uma plataforma de desenvolvimento de aplicativos Backend-as-a-Service (BaaS) que fornece serviços de backend hospedados, tais como banco de dados em tempo real, armazenamento em nuvem, autenticação, relatórios de falhas, aprendizado de máquina, configuração remota e hospedagem para seus arquivos estáticos.
- Entender como Instalar o Firebase
- Conhecer a documentação do Firebase
- Conhecer as ferramentas do Firebase disponíveis

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

- [ ] **Design Responsivo**:

- Ajustar suas páginas para o tamanho da tela do usuário
- Media queries
- Conhecer o conceito de Mobile first
