Guia Ágil

TechGuide - Alura

Front-end

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()'

- [ ] **Acessibilidade em Javascript**:

- Acessibilidade Digital é a eliminação de barreiras na Web. O conceito pressupõe que os sites e portais sejam projetados de modo que todas as pessoas possam perceber, entender, navegar e interagir de maneira efetiva com as páginas.
- Escrever código com acessibilidade em mente

- [ ] **Estratégias de SEO**:

- SEO significa otimização para motores de busca e que diz respeito as estratégias usadas para ranquear um site dentro de mecanismos de busca como o Yahoo, o Bing e, claro, o mais famoso de todos, o Google.
- Escolher palavras-chave
- Entender como o Google classifica páginas
- Conhecer os fatores de rankeamento
- Fazer Link Building

- [ ] **Design Responsivo**:

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

Nivel 2

- [ ] **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 - Testes**:

- O teste de software é o processo de avaliação e verificação de que um software realmente faz o que deveria fazer. Os benefícios dos testes incluem a prevenção de bugs, a redução dos custos de desenvolvimento e a melhoria do desempenho.
- Usar testes unitários
- Usar testes de integração
- Usar testes de comportamento (behavior)
- Usar mocks

☐ **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

- [ ] **JavaScript - ES6**:

- Conhecer as diferenças dessa versão do JavaScript

- [ ] **JavaScript - Modularização**:

- Isolar partes do código em módulos
- Usar import e export

- [ ] **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

- [ ] **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.

- [ ] **Cypress**:

- Criar e executar testes

Nivel 3

- [ ] **Estruturas de Dados**:

- No contexto dos computadores, uma estrutura de dados é uma forma específica de armazenar e organizar os dados na memória do computador para que esses dados possam ser facilmente recuperados e utilizados de forma eficiente quando necessário posteriormente.

- Conhecer as principais estruturas de dados
- Implementar as principais estruturas de dados

- [ ] **Conceitos de Orientação a Objetos**:

- A Programação Orientada a Objetos é um paradigma de programação de software baseado na composição e interação entre diversas unidades chamadas de 'objetos' e as classes, que contêm uma identidade, propriedades e métodos). Ela é baseada em quatro componentes da programação: abstração digital, encapsulamento, herança e polimorfismo.
- Como funcionam objetos
- Criar e utilizar construtores
- O que são classes
- Criar e utilizar métodos
- Como funciona encapsulamento
- O que é herança
- O que é polimorfismo
- Como funcionam interfaces
- O que são abstrações

- [ ] **JavaScript - Armazenamento**:

- Armazenar dados no front-end com localStorage
- Manipular dados armazenados
- Persistir dados armazenados

- [ ] **JavaScript - Concorrência**:

- Programação concorrente é um paradigma de programação para a construção de programas que fazem uso da execução simultânea de várias tarefas computacionais interativas, que podem ser implementadas como programas separados ou como um conjunto de threads criadas por um único programa.
- Executar tarefas paralelamente

- [ ] **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
- [ ] **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
- [ ] **Micro Frontends**:
  - Micro Frontends são uma abordagem arquitetural na qual uma aplicação web é dividida em partes independentes, cada uma representando uma parte específica da interface do usuário.
  - Divide uma aplicação web em partes independentes, facilitando o desenvolvimento e a manutenção.

- Possibilita a implantação e atualização contínua de partes específicas da aplicação sem afetar o aplicativo como um todo.
- Permite que equipes trabalhem de forma independente em diferentes partes da aplicação.
- Integra diferentes tecnologias e frameworks para desenvolver microfrontends, como React, Angular, Vue.js, entre outros.

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
