# Guia Ágil

TechGuide - Alura

## Full-stack

### 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

- [ ] **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
- [ ] **Node.js - Fundamentos**:
    - Node.js é um ambiente de execução JavaScript que permite executar aplicações desenvolvidas com a linguagem de forma autônoma, sem depender de um navegador.
    - Conhecer operações bloqueantes e não-bloqueantes
    - Aprender o conceito de laço de eventos (event loop)
    - Aprender a usar as bibliotecas do Node.js, como 'net', 'fs', 'http', 'path', entre outras
    - Entender como Timers funcionam
- [ ] **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

☐ **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()'

☐ **Criando uma aplicação React**:

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

☐ **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

☐ **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

- [ ] **Java - Fundamentos**:

- Java é uma linguagem de programação amplamente usada para codificar aplicações Web. Java é uma linguagem multiplataforma, orientada a objetos e centrada em rede que pode ser usada como uma plataforma em si. É uma linguagem de programação rápida, segura e confiável para codificar tudo, desde aplicações móveis e software empresarial até aplicações de big data e tecnologias do servidor.
- Conhecer os tipos primitivos
- Declarar variáveis, considerando os diferentes tipos
- 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 métodos
- Manipular arrays e listas
- Obter dados de uma API
- Criar construtores

- [ ] **Python - Fundamentos**:

- Python é uma linguagem de programação de alto nível, de uso geral, amplamente utilizada em aplicações web, desenvolvimento de software, ciência de dados e Machine Learning. Sua filosofia de projeto enfatiza a legibilidade do código com o uso de indentação significativa. Python é dinamicamente tipada e tem um garbage collector.
- Conhecer os tipos primitivos
- Declarar variáveis, considerando os diferentes tipos

- 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 métodos
- Manipular arrays e listas
- Obter dados de uma API
- Criar construtores
- Funções anônimas

☐ **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

☐ **Angular - Fundamentos**:

- Angular é uma framework de construção de aplicações e plataforma de desenvolvimento construído em TypeScript para criar aplicações eficientes e sofisticadas de página única (SPA).
- Construir interfaces utilizando HTML, CSS e TypeScript
- Criar aplicações SPA
- Construir aplicações web, mobile ou desktop
- Integrar dados com API's REST
- Utilizar a composição para criar componentes reutilizáveis
- Utilizar serviços do tipo Resolver
- Manipular requisições criando serviços do tipo Interceptor

## Nivel 2

- [ ] **Node.js - Express**:
  - O Express é um framework de aplicações web Node.js flexível que fornece um conjunto robusto de recursos para aplicativos da web e mobile.
  - Utilizar o framework Express para criação de APIs REST com Node.js
  - Gerenciar requisições de diferentes verbos HTTP em diferentes URLs
  - Definir a porta a ser usada para conexão e a localização dos modelos que são usados para renderizar a resposta
  - Criar manipuladores de rotas usando o método 'router'
  - Conhecer a bibliotecas 'Router' e seus verbos HTTP, como 'get', 'post', 'put', etc.
  - Definir endpoints com route paths
- [ ] **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
- [ ] **Node.js - ORM**:
  - Object-Relational Mapping (ORM), em português, mapeamento objeto-relacional, é uma técnica utilizada para fazer o mapeamento entre sistemas orientados a objetos e bancos de dados relacionais, onde as tabelas do banco de dados são representadas em classes e os registros das tabelas seriam instâncias dessas classes.
  - Entender o que são ORMs e para que são utilizados
  - Conhecer o SQL e seus gerenciadores de bancos de dados
  - Trabalhar com o Sequelize, um ORM para uso com Node.js

- Conhecer outros ORMs Node.js, como o Prisma

- [ ] **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 - 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

- [ ] **JavaScript - Modularização**:

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

- [ ] **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.

- [ ] **JavaScript - Armazenamento**:

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

- [ ] **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

- [ ] **Python - Comunicação com APIs**:

- Uma API é uma interface que desenvolvedores de software utilizam para programar a interação com componentes ou recursos de software fora de seu próprio código. Uma definição ainda mais simples é que uma API é a parte de um componente de software que é acessível a outros componentes.
- Entender o que é uma API REST
- Conhecer os comandos básicos de comunicação HTTP
- Entender o que é uma API REST
- Saber fazer requisições autenticadas
- Converter objetos para JSON e vice-versa
- Saber usar as ferramentas do pacote Requests

- [ ] **Spring Framework**:

- O Spring é um framework open source para a plataforma Java. Trata-se de um framework não intrusivo, baseado nos padrões de projeto (design patterns) de inversão de controle (IoC) e injeção de dependência. No Spring o contêiner se encarrega de "instanciar" classes de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do framework, o que é chamado de auto-wiring ou ainda anotações nas classes, métodos e propriedades. Dessa forma, o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.
- Entender o conceito de Injeção de Dependências

- Entender o padrão MVC
- Usar o Spring Data para manipular dados

- [ ] **Cypress**:

- Criar e executar testes

## Nivel 3

- [ ] **Arquitetura de Microsserviços**:

- Microsserviços são uma abordagem de arquitetura na qual o software consiste de pequenos serviços independentes que se comunicam entre si e são organizados de acordo com seus domínios de negócio.
- Aprender o conceito de arquitetura planejada para microsserviços
- Realizar a comunicação usando APIs
- Melhorar a escalabilidade de um sistema

- [ ] **Node.js - Autenticação e Tokens**:

- O JWT é um método criado pelo padrão RFC 7519 que representa a comunicação segura entre duas partes. Esse token é composto por três partes: cabeçalho, carga útil e assinatura.
- Construir um sistema de autenticação usando tokens
- Entender o funcionamento do JSON Web Token (JWT)
- Construir uma allowlist para guardar tokens opacos
- Implementar métodos de atualização de tokens

- [ ] **Contêineres**:

- Os contêineres são pacotes de software que contêm todos os elementos necessários para serem executados em qualquer ambiente. Gerenciamento de contêineres é uma área crucial na computação em nuvem e DevOps, que envolve o uso de tecnologias para automatizar o processo de criação, implantação, escalonamento e monitoramento de contêineres. Contêineres são unidades de software padronizadas que permitem aos desenvolvedores

empacotar todas as dependências de um aplicativo (código, bibliotecas, configurações, etc.) em um único pacote. Isso permite que o aplicativo seja executado de forma consistente em qualquer ambiente de infraestrutura.

- A tecnologia de contêineres, como exemplificada pelo Docker, fornece um ambiente consistente e portátil para desenvolvimento, teste e implantação de aplicativos, o que é vital para o trabalho eficiente de engenharia de dados. Além disso, o Kubernetes, um sistema de orquestração de contêineres, permite o gerenciamento, a automação e a escalabilidade de aplicações baseadas em contêineres em ambientes de produção. Dominar esses conceitos e tecnologias possibilita a engenheiros de dados construir e manter pipelines de dados eficientes e confiáveis.
- O Kubernetes (também conhecido como k8s ou kube) é uma plataforma de orquestração de containers open source que automatiza grande parte dos processos manuais necessários para implantar, gerenciar e escalar aplicações em containers.
- Isolar seu software para funcionar independentemente
- Implantar software em clusters
- Modularizar seu sistema em pacotes menores
- Conhecer a plataforma Docker
- Conhecer Kubernetes

☐ **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

- [ ] **WebSockets**:
    - WebSocket é uma tecnologia que permite a comunicação bidirecional por canais full-duplex sobre um único soquete Transmission Control Protocol (TCP). Ele é projetado para ser executado em browsers e servidores web que suportem o HTML5, mas pode ser usado por qualquer cliente ou servidor de aplicativos.
    - Conhecer o protocolo WebSocket e seu uso na comunicação cliente-servidor
    - Aprender sobre os diversos usos do WebSocket na web
    - Criar aplicações que utilizam WebSockets com as APIs e bibliotecas do Node.js
- [ ] **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
- [ ] **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

- [ ] **Nest.js - Fundamentos**:
    - NestJS é um framework Node com total suporte a TypeScript e que roda sobre frameworks HTTP como expressJS ou Fastify. Ele utiliza diversos elementos de programação orientada a objetos e uma série de funcionalidades do TypeScript.
    - Aprender o que é o NestJS e por que é utilizado
    - Utilizar recursos específicos do NestJS, como providers, módulos e controllers
    - Desenvolver APIs usando o NestJS
    - Usar a Interface de Linha de Comando (CLI) do Nest.js
- [ ] **Kafka**:
    - O Apache Kafka é uma plataforma distribuída de transmissão de dados que é capaz de publicar, subscrever, armazenar e processar fluxos de registro em tempo real. Essa plataforma foi desenvolvida para processar fluxos de dados provenientes de diversas fontes e entregá-los a vários clientes.
    - Utilizar o Kafka para comunicação assíncrona
    - Criar microsserviços com Kafka
    - Criar produtores e consumidores
    - Entender como usar o Kafka para paralelismo e execução serializada
    - Obter garantias relativas ao envio ou entrega das mensagens

## Habilidade Auxiliar: Infraestrutura e boas práticas

- [ ] **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

- [ ] **Entrega e integração contínuas (CI/CD)**:

- CI/CD é a abreviação de Continuous Integration/Continuous Delivery, traduzindo para o português "entrega e integração contínuas". Trata-se de uma prática de desenvolvimento de software que visa tornar a integração de código mais eficiente por meio de builds e testes automatizados.
- Automatizar a integração de código entre varias partes da equipe se tornou cada vez mais importante, ja que assim é possivel acelerar o

desenvolvimento e diminiur o tempo de entrega de software.

- Executar testes automatizados da aplicação para verificar seu funcionamento.
- Realizar a entrega de atualizações de forma automatica e com segurança.
- Realizar testes de conexão e testes de carga para evitar que a aplicação apresente problemas ao ser atualizada.

☐ **SQL - Fundamentos**:

- SQL (Structured Query Language, traduzindo, Linguagem de Consulta Estruturada) é uma linguagem de programação padronizada que é usada para gerenciar bancos de dados relacionais e realizar várias operações sobre os dados neles contidos.
- Conhecer os comandos mais comuns do SQL
- Usar SELECT para consultar uma tabela
- Usar INSERT para inserir dados em uma tabela
- Usar UPDATE para atualizar uma tabela
- Usar DELETE para remover dados de uma tabela
- Usar JOIN para conectar os dados de múltiplas tabelas
- Conhecer as cláusulas (FROM, ORDER BY, etc)

☐ **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.

☐ **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

- [ ] **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.
- [ ] **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
- [ ] **Conceitos de Design Orientado a Domínio (Domain-Driven Design - DDD)**:
  - O Design Orientado a Domínio (DDD) é uma abordagem ao projeto e desenvolvimento de software que é primeiramente informado pelos requisitos de negócios. Os componentes do programa (objetos, classes, matrizes, etc.) indicam a indústria, setor ou domínio empresarial em que o negócio opera.
  - Modelar domínios de forma efetiva
  - Basear projetos complexos em modelos do domínio
  - Conhecer os blocos de construção de DDD

## Habilidade Auxiliar: UX e Design

- [ ] **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
- [ ] **Design Responsivo**:
  - Ajustar suas páginas para o tamanho da tela do usuário
  - Media queries
  - Conhecer o conceito de Mobile first
- [ ] **Componentes de design**:
  - Conhecer os componentes descrevem um layout ou interface
- [ ] **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
- [ ] **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

---