Angular

TechGuide - Alura, FIAP e PM3

Angular

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

☐ **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

☐ **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

☐ **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

☐ **RxJS - Fundamentos**:

- A RxJS é uma biblioteca para a composição de programas assíncronos e baseados em eventos, utilizando sequências observáveis. Ela fornece o tipo núcleo, o Observable, tipos satélite (Observer, Schedulers, Subjects) e operadores inspirados nos métodos Array (map, filter, reduce, every, etc.) para permitir a manipulação de eventos assíncronos como coleções.

- Criar programas assíncronos
- Criar programas baseados em eventos
- Entender o conceito de observables e sequências de observables
- Entender como usar Observers, Subscription, Subject

- [ ] **O pattern Observer**:

- Em engenharia de software, o padrão (pattern) chamado Observer é um padrão de projeto de software no qual um objeto, chamado de sujeito (subject), mantém uma lista de seus dependentes, chamados de observadores (observers), e os notifica automaticamente de qualquer mudança de estado, geralmente chamando um de seus métodos.
- Entender o que são Design Patterns
- Atualizar diversos elementos simultaneamente usando Observers
- Declarar os Subjects
- Criar programas baseados em eventos

Nivel 2

- [ ] **Angular - Templates**:

- Em Angular, templates são um modelo para um fragmento de uma interface de usuário (IU). Templates são escritos em HTML, e uma sintaxe especial pode ser usada dentro de um template para construir com base em muitas das características do Angular.

- [ ] **Angular - Renderização**:

- Uma aplicação Angular normal é executada no navegador, renderizando páginas no DOM em resposta às ações do usuário. A Angular Universal executa no servidor, gerando páginas de aplicação estática que mais tarde são inicializadas no cliente.
- Exibir uma página Angular no navegador
- Realizar renderização no lado do servidor

- [ ] **Angular - Services**:
  - Um serviço é uma categoria ampla que engloba qualquer valor, função ou funcionalidade que uma aplicação necessite. Um serviço é tipicamente uma classe com um propósito conciso e bem definido. O Angular distingue componentes de serviços, para aumentar a modularidade e a reusabilidade.
  - Criar dados que estarão sempre disponíveis
  - Dividir a aplicação web em diversas partes
- [ ] **Angular - Roteamento**:
  - Navegar até um componente
  - Incluir um parâmetro de rota
  - Controlar o fluxo de navegação do seu usuário com guarda de rotas
- [ ] **Angular - CLI (Interface de Linha de Comando)**:
  - A Interface de Linha de Comando (CLI) do Angular é uma ferramenta de interface de linha de comando que você utiliza para inicializar, desenvolver, estruturar e manter aplicações Angular diretamente de um shell de comando.
  - Aprender a sintaxe 'ng [argumento-opcional] [opções]'
  - Conhecer os comandos mais importantes, como 'ng add', 'ng build', 'ng update', 'ng deploy', 'ng new', 'ng test', entre outros

Nivel 3

- [ ] **Angular - Gerenciamento de Estado**:
  - Um 'estado' (state) é qualquer dado necessário para reconstruir UI (Interface de Usuário) qualquer momento. A alteração desses dados desencadeará um redesenho da interface do usuário. O gerenciamento do estado é o conceito de adicionar, atualizar, remover e ler esses dedos e seus 'estados' em uma aplicação.
  - Atualizar componentes em tempo real
  - Esperar por alterações em algum componente e executar alterações

- Usar Redux, NGXS e outros

- [ ] **Angular - Formulários**:

- Um formulário web é uma página online que aceita inseção de dados (inputs) do usuário. É uma página interativa que imita um documento ou formulário em papel, onde os usuários preenchem determinados campos.
- Criar formulários com Template Forms
- Criar formulários reativos com Reactive Forms

- [ ] **Angular - Módulos**:

- Um módulo (Module), diferentemente de um componente, não controla nenhuma view. Um módulo é constituído de um ou mais componentes. Uma aplicação Angular tem que ter no mínimo um módulo que contenha mínimo um componente (Component).
- Como usar a classe ngModule
- Declarar quais componentes podem ser usados por componentes de outros módulos
- Declarar quais services serão injetados
- Aprender a modularizar uma aplicação
- Carregar módulos como Lazy

- [ ] **Angular - Injeção de Dependências**:

- A injeção de dependência permite declarar as dependências de suas classes TypeScript sem cuidar de sua instanciação. Em vez disso, o Angular trata da instanciação para você. Este design pattern permite escrever um código mais testável e flexível.
- Declarar as dependências de suas classes
- Injetar dependências em um componente
- Conhecer os Injection Tokens
- Configurar providers
- Entender a Injeção de Dependência Hierárquica

- [ ] **Angular - 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
  - Familiarizar-se com Jasmine e/ou Karma
- [ ] **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
- [ ] **Angular - Design System**:
  - Um Design System é uma coleção organizada de componentes, padrões e diretrizes que ajudam equipes a criar interfaces consistentes e escaláveis. Ele atua como um guia para o design e desenvolvimento de produtos digitais, garantindo uniformidade e eficiência. No Angular, o Design System ganha ainda mais poder quando implementado em um monorepo e documentado com o Storybook.

- Criar um Design System com Angular;
- Organizar o projeto em um Monorepo;
- Usar o Storybook para documentar e testar os componentes criados.

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

☐ **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

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
