Vue

TechGuide - Alura, FIAP e PM3

Vue

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

- [ ] **Vue - Componentes**:

- O Vue permite definir componentes usando a Option API ou a Composition API.
- Para que servem e como funcionam componentes
- Conheça a especificação da Sintaxe SFC (Single File Component)

- [ ] **Vue - 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

- [ ] **Vue - Diretivas**:

- Explore as diretivas básicas oferecidas pelo Vue, como v-if, v-else, v-show, v-bind, v-model e v-for.
- Depois de se familiarizar com as diretivas integradas, suba de nível criando suas próprias diretivas personalizadas.
- Para finalizar sua jornada, domine os argumentos e modificadores das diretivas.

- [ ] **Vue - Options API**:
  - Aprenda a estruturar um componente Vue usando a Options API, abrangendo opções como 'data', 'methods', 'computed', e 'watch'.
  - Domine os hooks do ciclo de vida do componente ('created', 'mounted', 'updated', e 'destroyed') para controlar comportamentos em diferentes etapas da vida do componente.
  - Pratique o gerenciamento de estado reativo dentro dos componentes, utilizando opções como 'data' e 'computed', e aprenda como sincronizar estados entre componentes usando 'props' e 'emit'.
- [ ] **Vue - Composition API**:
  - Inicie componentes com a função setup(). Ela é o coração da Composition API.
  - Aprenda a usar ref e reactive para criar e gerenciar estados reativos.
  - Domine os hooks do ciclo de vida. Eles são vitais para controlar comportamentos em diferentes estágios do componente.
- [ ] **Vue - Template**:
  - Comece aprendendo a sintaxe básica de um template Vue e como renderizar dados dinâmicos.
  - Aprofunde-se no uso de diretivas de template como v-if, v-for e v-bind para criar interfaces dinâmicas.
  - Aprenda a definir e usar componentes dentro dos seus templates para reutilizar e organizar o código.
- [ ] **Vue - Create App**:
  - Inicie criando um novo projeto Vue com Vue CLI ou Vite.
  - Crie componentes Vue e entenda como estruturá-los usando Options API ou Composition API.
  - Veja como a navegação com Vue Router e gerenciamento de estado com Pinea podem ser úteis em SPAs escritas em Vue.
- [ ] **Vue Forms**:

- Comece aprendendo as bases dos formulários no Vue, incluindo v-model e como vincular diferentes tipos de campos de entrada (input binds).
- Depois de dominar as bases, aprenda como o Vue 3 lida com as entradas do usuário e as diferenças em relação às versões anteriores.
- Mergulhe no mundo da Vue Formulate, uma das bibliotecas mais completas para trabalhar com formulários no Vue. Compreenda como usar seus recursos para construir formulários complexos com facilidade.
- Continue sua jornada com a VeeValidate, uma biblioteca para validação de formulários que oferece muitas opções de validação e permite que você crie suas próprias regras.
- Explore a Vuelidate, outra biblioteca de validação poderosa que oferece uma abordagem um pouco diferente da VeeValidate.
- Além de usar bibliotecas, aprenda a criar suas próprias regras de validação personalizadas no Vue.
- Aprofunde-se na criação de formulários dinâmicos, onde os campos podem mudar com base nas ações do usuário.
- Descubra como integrar a gestão dos formulários com Vuex/Pinia para um gerenciamento de estado mais sólido e escalável.

☐ **Vue - Eventos**:

- Inicie seu aprendizado sobre eventos no Vue, entendendo a diferença entre eventos do DOM e eventos personalizados do Vue.
- Aprenda a usar a diretiva v-on para adicionar manipuladores de eventos a elementos do DOM e como passar argumentos para os manipuladores de eventos.
- Mergulhe nos eventos de teclado e mouse, entendendo como capturá-los e como usar modificadores de eventos para simplificar o código.
- Descubra como emitir e ouvir eventos personalizados usando o método $emit e a diretiva v-on.
- Aprofunde-se na compreensão do bubbling e capturing de eventos no Vue, e quando usar o .native modifier.

- Conheça o conceito de "Event Bus" para comunicação entre componentes não relacionados.

- [ ] **Vue Router**:

- Comece instalando e configurando o Vue Router em seu projeto. Aprenda a criar um arquivo de rotas e a configurar o Vue para usar o Vue Router.
- Domine a criação de rotas básicas, entenda a correspondência de rotas com os componentes Vue e como usar a diretiva para criar links.
- Avance para o roteamento dinâmico, aprendendo a usar parâmetros de rota para criar rotas flexíveis.
- Entenda o roteamento aninhado e como usar o componente para renderizar componentes de rota aninhados.
- Mergulhe no roteamento nomeado e aprenda como ele pode tornar o seu código mais fácil de entender e manter.
- Explore os guardas de rota e como eles podem ser usados para controlar o acesso a certas rotas.
- Aprenda a criar transições suaves entre as rotas com o elemento do Vue.

Nivel 2

- [ ] **Vue - Composition API a fundo**:

- Comece entendendo a função setup() e o conceito de reatividade no Vue 3.
- Domine ref e reactive, os principais elementos para gerenciar estados reativos.
- Aprenda a usar computed para trabalhar com propriedades calculadas.
- Explore o uso de watch e watchEffect para reagir às mudanças de estado.
- Entenda os hooks do ciclo de vida para controlar o comportamento do componente em diferentes fases.
- Crie e exponha funções a partir da função setup.
- Aprenda a usar props dentro da setup e como lidar com as diretivas no template.

- Implemente a lógica reutilizável através de Composables.
- Aprenda as melhores práticas e padrões comuns ao usar a Composition API.

- [ ] **Vue Devtools**:

- Comece instalando a extensão Vue DevTools no seu navegador.
- Familiarize-se com a interface, aprendendo sobre as abas de Componentes, Vuex, Roteamento e Performance.
- Aprenda a inspecionar componentes, observando seus props, dados, e slots.
- Entenda como monitorar a store do Vuex | Pinia e o histórico de roteamento.
- Pratique a edição no local do estado da aplicação e o "time-travel debugging".
- Ajuste as configurações ao seu gosto e integre as DevTools com seu ambiente de desenvolvimento.

- [ ] **Vue - Estilizando componentes**:

- Comece pelo básico, aplicando folhas de estilo CSS externas ao seu projeto Vue.
- Vá um pouco mais fundo e aprenda sobre o atributo "scoped" no Vue, que permite que você direcione estilos especificamente para um único componente.
- Explore bibliotecas que permitem que você escreva estilos em JavaScript, oferecendo recursos dinâmicos e reutilizáveis.
- Veja como os pré-processadores CSS, como SASS ou LESS, podem ser integrados ao seu projeto Vue para estilização avançada.
- Experimente bibliotecas populares de componentes Vue, como Vuetify ou BootstrapVue, que vêm com seus próprios conjuntos de estilos e temas.
- Conheça a biblioteca vue-styled-components, uma implementação do Styled Components para Vue, que permite escrever CSS real em JavaScript.
- Dê uma olhada no Tailwind CSS, um framework de CSS de utilidade, e aprenda como pode ser integrado com Vue para um desenvolvimento de interface de usuário eficiente.

- Com o Vue 3, explore a vinculação de estilos e como eles podem ser usados para aplicar estilos dinâmicos com base nos dados do componente.

- [ ] **Vue - Comunicando-se com APIs**:

- Entenda o que são APIs, como funcionam e como se comunicam com os clientes web.
- Aprenda os conceitos básicos de AJAX e como o Vue se integra com AJAX para recuperar dados de uma API.
- Mergulhe no Axios, uma popular biblioteca JavaScript para realizar solicitações HTTP. Aprenda a instalar e a usar o Axios em um projeto Vue para fazer chamadas GET, POST e outras.
- Conheça o unfetch, uma alternativa leve e moderna para fazer chamadas de API. Veja como ele pode ser usado em um projeto Vue.
- Adentre no mundo do GraphQL, um poderoso paradigma de API. Aprenda a usar o Apollo Client, uma solução completa para o gerenciamento de estado de dados GraphQL em aplicações Vue.
- Explore o Vue Relay, uma estrutura para trabalhar com GraphQL no Vue. Entenda as vantagens que ele oferece e como ele difere do Apollo Client.
- Aprenda a lidar com as respostas das APIs, incluindo a manipulação de dados recebidos e a tratamento de erros.

- [ ] **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

- [ ] **Vue - Testes**:

- Testes unitários são testes que validam a funcionalidade de partes individuais do seu código.
- Testes E2E simulam a experiência do usuário ao interagir com a aplicação, garantindo que todos os componentes funcionem juntos como esperado.
- Vue Test Utils é a biblioteca oficial para testes de componentes Vue.
- Aprenda a escrever testes básicos para seus componentes Vue, verificando se eles são renderizados corretamente.

- Os métodos dos componentes são partes cruciais do código que muitas vezes contêm lógica complexa. Aprenda a testá-los efetivamente.
- A injeção de dependência pode facilitar o teste de partes do seu código que dependem de serviços ou outros componentes.
- Se você estiver usando Vuex, Pinia ou Vue Router em seu aplicativo, aprender a lidar com eles nos testes é crucial.
- Cypress é uma ferramenta comum para realizar testes E2E em aplicações web, incluindo Vue. Instale e configure o Cypress para o seu projeto.
- Comece a escrever testes que simulam ações do usuário, como clicar em botões, preencher formulários e navegar entre as páginas.
- Os comandos personalizados ajudam a simplificar os testes, evitando a repetição de códigos de teste.
- Aprenda a manipular e testar diferentes estados da sua aplicação, como usuários logados e deslogados.
- Integre seus testes E2E ao seu processo de integração contínua/deploy contínuo (CI/CD) para garantir que sua aplicação esteja sempre funcionando como esperado.

☐ **Cypress**:

- Criar e executar testes

☐ **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

☐ **Vue - Performance**:

- Aprenda a usar ferramentas de profiling, como a Vue Devtools e o Performance tab no Chrome Devtools, para entender onde seu aplicativo pode estar sofrendo em termos de desempenho.
- Estude técnicas de otimização de carga de página, como lazy-loading, prefetching e preloading.
- Compreenda a importância de escolher a arquitetura certa para a sua aplicação Vue. Aprenda sobre padrões de projeto e como eles podem afetar a performance.
- Familiarize-se com as técnicas de redução do tamanho do bundle e tree-shaking para remover código desnecessário da sua aplicação.
- Aprenda a implementar a divisão de código para carregar apenas o código necessário para o usuário em um determinado momento.
- Aprofunde-se em técnicas de otimização de atualizações, como imutabilidade, estabilidade de props e uso de chaves em listas para minimizar re-renderizações desnecessárias.
- Entenda como garantir a estabilidade de props para evitar atualizações desnecessárias dos componentes.
- Familiarize-se com as diretivas v-once e v-memo e quando utilizá-las para otimizar a renderização de componentes.
- Aprenda sobre otimizações gerais como a virtualização de listas grandes, reduzindo a sobrecarga de reatividade para grandes estruturas imutáveis, e evitando abstrações de componentes desnecessárias.
- Entenda como a virtualização pode ser usada para exibir eficientemente grandes listas de dados, reduzindo a sobrecarga de renderização.

☐ **Vue - Componentes assíncronos**:

- Compreenda o que são e por que usar componentes assíncronos no Vue.
- Aprenda a criar um componente assíncrono básico usando a função defineAsyncComponent.
- Domine a renderização de um estado de loading enquanto o componente assíncrono é carregado.
- Implemente o tratamento de erros ao carregar o componente assíncrono.

- Aplique lazy loading nos seus componentes para melhorar a performance da aplicação.
- Use componentes assíncronos com Vue Router para carregar rotas sob demanda.

- [ ] **Vue - Teleporte**:

- Entenda o que é o Teleport e qual problema ele resolve no Vue.
- Aprenda a usar o componente para renderizar conteúdo em um local diferente do DOM.
- Domine a propriedade 'to' do Teleport, usada para especificar o local de destino no DOM.
- Explore como usar diretivas como v-if para controlar a renderização do conteúdo teleportado.
- Experimente casos de uso avançados, como teleportar modais, tooltips, notificações e muito mais.

- [ ] **Vue Slots**:

- Comece compreendendo o conceito de slots e como eles são usados para criar layouts personalizáveis e reutilizáveis em componentes Vue.
- Aprenda a usar slots anônimos, que são os slots padrão que você usa quando deseja inserir conteúdo personalizado no componente.
- Mergulhe em slots nomeados, que permitem que você personalize diferentes partes do layout do componente.
- Domine os scoped slots, que permitem o acesso aos dados do componente filho dentro do slot.
- Familiarize-se com a diretiva v-slot e como usá-la para criar slots nomeados e scoped slots.
- Descubra como usar slots dinâmicos para criar layouts ainda mais flexíveis.
- Entenda a diferença entre a nova sintaxe de slots no Vue 3 e a antiga no Vue 2 para garantir a compatibilidade com versões anteriores.

- [ ] **Quasar Framework**:

- Entenda o que é o Quasar e como ele se encaixa no ecossistema Vue.
- Configure o Quasar em seu ambiente de desenvolvimento local e aprenda como criar um novo projeto Quasar.
- Aprenda a usar os componentes pré-fabricados do Quasar, que são otimizados para eficiência e personalização.
- Domine o sistema de layout e roteamento do Quasar para construir aplicativos complexos e interativos.
- Aprenda a gerenciar o estado de sua aplicação Quasar com Vuex, a biblioteca de gerenciamento de estado do Vue.js.
- Conheça a CLI do Quasar, uma ferramenta poderosa para gerar, desenvolver e gerenciar seus projetos Quasar.
- Conheça as diferentes opções para o deploy de aplicações feitas com Quasar.

☐ **Nuxt**:

- Entenda o que é Nuxt.js, como ele funciona e quando usá-lo.
- Aprenda como instalar e configurar um projeto Nuxt.js a partir do zero.
- Familiarize-se com a estrutura de pastas de um projeto Nuxt.js e o que cada pasta faz.
- Compreenda a geração automática de rotas em Nuxt.js e como trabalhar com rotas dinâmicas.
- Aprenda a integrar soluções de gerenciamento do estado global em uma aplicação Nuxt.js.
- Entenda as diferenças entre SSR e SSG e como usá-los em Nuxt.js.
- Aprenda a usar plugins Nuxt.js para adicionar funcionalidades globais à sua aplicação.
- Explore como o middleware pode ajudá-lo a executar código antes de renderizar a página.
- Conheça as diferentes opções de deploy de uma aplicação Nuxt.js.

☐ **Pinia**:

- Aprenda os conceitos básicos do Vuex, o problema que ele resolve e quando usá-lo.
- Compreenda como o estado é armazenado no Vuex e como acessar o estado global em seus componentes Vue.
- Aprenda a usar getters para acessar valores derivados do estado.
- Descubra como realizar ações assíncronas através de actions.
- Aprenda como usar a store fora de componentes com o hook useStore
- Explore os guias de migração para mudar do vuex para o Pinia
- Mergulhe no universo do SSR combinando o poder do Pina com o Nuxt

☐ **Vue - Provide / Inject**:

- Compreenda o que são e por que você pode precisar deles. Eles são usados para injetar dependências em componentes descendentes sem passar props em todos os níveis.
- Aprenda a usar a opção provide em um componente para fornecer dados que podem ser injetados em componentes descendentes.
- Veja como a opção inject é usada em um componente descendente para acessar dados fornecidos por um componente ancestral.
- Descubra como o Provide / Inject funciona com a Composition API, usando provide e inject funções no método setup.
- Saiba como reagir às mudanças nos valores fornecidos usando um objeto reativo ou ref.
- Aprenda a lidar com situações em que vários ancestrais fornecem a mesma chave.
- Embora seja uma ferramenta poderosa, o Provide / Inject deve ser usado com moderação. Entenda quando é apropriado usá-lo e quando é melhor evitar.

☐ **Vue - Suspense**:

- O Suspense é um componente embutido para orquestração de dependências assíncronas em uma árvore de componente. Ele pode interpretar um estado de carregamento enquanto espera por várias

dependências assíncronas encaixadas em baixo da árvore de componente ser resolvida.

- Conhecer dependências assíncronas

- [ ] **NativeScript-Vue**:

- O NativeScript-Vue é um plugin NativeScript que permite usar Vue.js para criar seu aplicativo móvel.
- Criar apps com interface nativa

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

- [ ] **Vite**:

- Instalação: Instale o Vite globalmente e crie seu projeto Vue com um simples comando create-vite
- Explore a estrutura de diretórios do Vite e entenda como organizar seus componentes e rotas.
- Aprenda a usar os comandos de desenvolvimento e build do Vite para executar e otimizar sua aplicação.

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
