CONHECIMENTOS GERAIS

Então vamos lá, começando pelos conhecimentos GERAIS que absolutamente todas as áreas vão requisitar, então você PRECISA aprender...

Versionamento de código com GIT/GitHub

Saber fazer o controle básico de código é conhecimento obrigatório hoje em dia, pois até pra compartilhar códigos com seus amigos, você vai ser solicitado "poe no github e me passa o link".

Conhecimentos básicos de terminal

Mesmo que você use Windows, alguns conhecimentos básico de terminal/prompt são essenciais, pois hoje em dia, independente da linguagem que você escolher, uma hora ou outra vai ser necessário fazer uso do terminal/prompt, então o conhecimento básico dos principais comandos já é mais do que o suficiente.

Nomenclatura básica de programação

Coisas como API, padrão de projeto, encoding, licença, estrutura de dados, algoritmo, dentre outros, são nomenclaturas que você precisa no mínimo saber do que se trata. Não precisa saber tudo e todas as variações, mas pelo menos uma noção básica pra saber o que é quando ver a palavra em algum lugar (ou escutar).

FRONTEND

Beleza, agora que você já tem a orientação básica sobre o que precisa aprender antes de começar a brincar em qualquer área. Vamos aprender agora desde o início, qual é o melhor caminho para um programador Frontend que está começando agora.

HTML e CSS

Hoje em dia a gente ainda chama de HTML5 e CSS3, mas o fato é que esses números de versões já são o padrão mundial, então podemos apenas chamar de HTML e CSS e pronto.

Nessa tecnologia, o que você precisa focar mais?

- **Aprenda o básico de HTML e CSS.**
- **Aprenda o que é e como funciona HTML semântico.**
- **Aprenda SEO básico e tags de acessibilidade.**
- **Sobre CSS, foque em aprender, além do básico, Display, Positioning, Float e FlexBox.**
- **Depois foque em Media Query, pra tornar suas páginas responsivas.**
- **Por último, aprenda conceitos mais avançados de CSS3, como animações, before/after, transform, etc...**

Durante esse processo, é legal que você vá praticando tudo que vai aprendendo. Como? Criando e re-criando páginas que você conhece, bem como criando novas páginas.

Pesquisa lá: "website template", ou "layout de site", ou variações disso. Depois vai em Imagens e pronto, você terá uma centena de sites e páginas pra praticar recriando elas.

JavaScript

Essa é a linguagem-mãe de quase todas as tecnologias que você vai manusear quando o assunto é frontend, então eu recomendo dar uma atenção especial nos estudos aqui.

O que você deve focar mais atenção pra aprender bem?

- **Sintaxe e construções básicas.**
- **Manipulação de DOM**
- **Eventos diversos (mouse e teclado)**
- **Fetch API e Ajax**

Só com esse conhecimento, além do HTML e CSS, você não faz ideia da habilidade que já vai ter, já vai ser capaz de criar coisas que antes eram inimagináveis pra você conseguir criar um dia.

A partir daí, foca em **EcmaScript 6+**, **Javascript Modular**, **Controle de Eventos**, **Strict**, **DNS**, **Hoisting**, etc.

Package Managers

NPM

Yarn

Agora você vai começar a sentir o poder do Javascript, então o ideal é já aprender todas as principais funções e comandos dos dois maiores gerenciadores de pacotes para Javascript... NPM e YARN. Os dois usam os mesmos repositórios, o mais usado é o NPM, mas o Yarn tem recursos mais otimizados, se tornando mais ágil... Nada sério, escolha qualquer um e seja feliz.

Sass

Bootstrap

Pré-processador de CSS

As 3 maiores opções do mercado hoje em dia são SASS, LESS e POSTCSS, sendo a última não considerada um pré-processador, mas sim um pós-processador, não importa, depois você estuda sobre essa diferença.

Apesar de ainda existir mercado pro LESS, o mundo hoje usa majoritariamente o SASS, então essa é minha recomendação.

Gulp

ESLint

Webpack

Prettier

Framework CSS

Existem 2 grandes no mercado, Bootstrap e Materialize, sendo o primeiro disparadamente maior. Então pra Framework CSS hoje, minha recomendação é Bootstrap. Aprenda e pratique muito, pois vai ficar viciado em usar ele.

Linters, Bundlers e Taskers

Você vai aprender agora sobre ESLint e Prettier, ferramentas que vão te ajudar a escrever um código mais limpo e organizado. Logo em seguida, estude duas ferramentas excepcionais, Webpack e Gulp. Com elas, você vai conseguir organizar seu projeto e automatizar tarefas de forma muito rápida.

Se for aprender React, não esqueça de aprender **Hooks** e **Redux**, confie em mim.

React

Se for aprender Vue, também não esqueça do **VueX**.

Vue

No caso do Angular, a recomendação pra aprofundamento vai para a ferramenta **RxJS**.

Angular

Frameworks

A minha maior recomendação é que você experimente todas as 3 tecnologias até conseguir criar, por exemplo, um sisteminha de tarefas em cada uma. Aí sim você vai ter condições de decidir na prática qual gostou mais.

O React hoje possui um grande aliado, o React Native, que te permite criar aplicativos nativos usando códigos com React, ou seja, é um conhecimento que você vai conseguir pelo menos 2 aplicações reais, tanto no Frontend quanto no Mobile.

Então em termos de Vagas de Emprego, procura de mercado e claro, preferência pessoal, a minha recomendação dentre os três vai ser para o React. Apesar do Vue também ter um excelente mercado e muita gente achar ele bem mais fácil de aprender.

Styled Components

Essa ferramenta é muito conhecida no mundo React, mas saiba que ela também é compatível com os outros frameworks.

CSS in JS

Cada vez mais hoje em dia nós estamos usando código CSS dentro do nosso Javascript, o que faz com que muita gente fique p* da vida com a junção, mas é um caminho sem volta, então se acostume e aprenda, pois vai te ajudar bastante.

Jest

Cypress

Enzyme

TESTING

Agora você já tem um conhecimento muito bom no Frontend, tá na hora de avançar um pouco isso, então vá estudar sobre Testing.

Web API e PWA

O próximo passo é aprender as principais WebAPIs, como por exemplo:

Storage

Location

WebSockets

ServiceWorkers

Notification

E claro, todo o básico sobre PWA e suas aplicações práticas.

Pra não deixar de citar, você pode aprender **TypeScript** também, pra dar um upgrade nos seus conhecimentos de Javascript.

BACKEND

Legal, você decidiu se especializar em Backend, vamos falar sobre os caminhos possíveis, minhas recomendações e todas as variações de escolha.

Hoje em dia quando se fala em Backend, nós logo pensamos em criação de **APIs**.

Claro, Backend não se resume a criar APIs, mas hoje em dia essa é uma grande habilidade de programadores Backend.

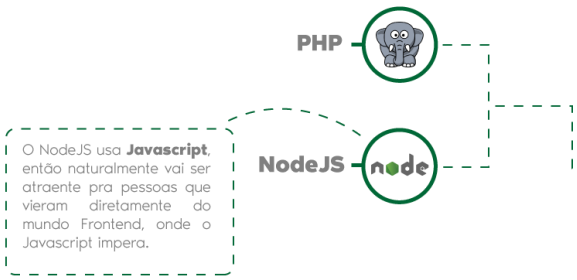
Quando você estiver estudando, uma das grandes técnicas de aprendizado é pegar um layout já feito e criar o backend dele, ou fazer ele funcionar no servidor

Linguagens Backend

Hoje você tem 5 grandes opções de escolha pra linguagens backend, que cada um vai te dar um mar de escolhas subsequentes.

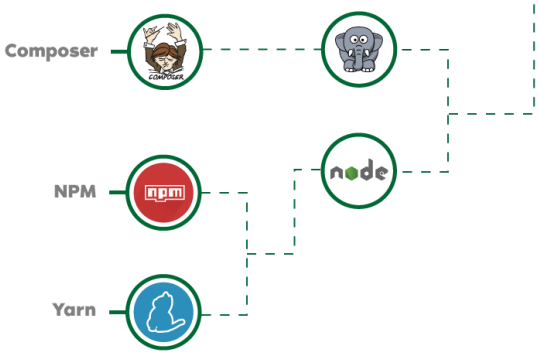
Tem outras, mas essas são as principais atualmente, embora pessoalmente eu só recomende 2 dessas pra backend hoje em dia, **PHP** ou **NodeJS**, com base em um balanço geral entre emprego, uso no mercado e performance.

Escolha da Linguagem



Para iniciantes, escolha **PHP** ou **NodeJS.** Vamos agora para as peculiaridades de cada uma

Gerenciadores de pacotes



Você precisa aprender sobre os gerenciadores de pacotes da tecnologia que você escolher.

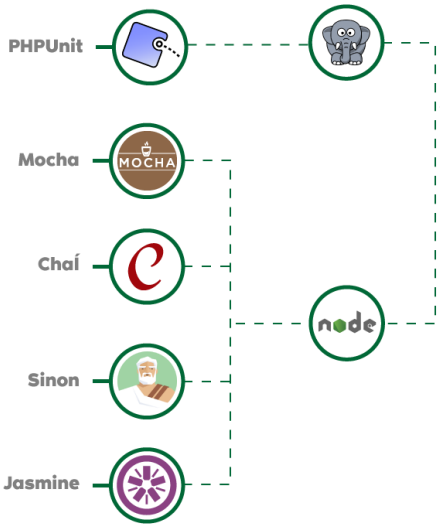
Padronizações e melhores práticas

Aprenda as padronizações e melhores práticas de cada uma, por exemplo..

O **PHP** tem o **PHP-FIG** e as **PSRs**, que vão te dar excelentes orientações sobre como criar seu código, como organizar, como arquitetar algumas coisas específicas.

O **Node** é menos padronizado quanto a isso, pois há vários métodos de padronização diferentes, mantidos pela comunidade, mas claro, as boas práticas aprendidas no Javascript vão ser muito úteis.

Aprenda sobre Testing



Aprenda a manipular bancos de dados relacionais.

Isso mesmo, independente de ter escolhido PHP ou Node, você deve ter conhecimentos em **bancos de dados relacionais**

Hoje em dia, banco de dados é a vida de qualquer sistema, pois é onde você vai armazenar informações que vão tornar seu sistema dinâmico.

Agora vamos pra prática

Usando tudo que você aprendeu até agora, crie um **Blog** completo, com diferentes autores, em que cada autor pode se registrar, entrar na própria conta, criar, atualizar e deletar seus próprios posts, mas não os posts dos outros.

O nome desse conceito é **CRUD** *(Create, Read, Update, Delete)*, ou seja, um sisteminha capaz de fazer as 4 principais operações na programação, que são a base pra qualquer sistema cadastral.

MongoDB

hoje em dia o **maior** e **melhor** framework do mercado, então com toda certeza vale a pena aprender ele, todos os seus detalhes, regras, aplicabilidades e funcionalidades.

Laravel

Express

Já vem com compatibilidade à um mundo de bibliotecas que podem ser usadas através do gerenciador de pacotes, o que acaba tornando um framework simples, em um dos mais usados do mercado hoje em dia.

Apache

Nginx

Se você está aprendendo Node, ele mesmo já cria um servidor pra ti, então esse aprendizado vai ser quase que uma consequência natural dos estudos.

Docker

Para criar um ambiente flexível e portátil

Aprenda a manipular bancos de dados não relacionais.

Você deve ter conhecimento tanto em bancos relacionais quanto não relacionais, nesse caso, a melhor opção disparado é o **MongoDB**. Principalmente se você está estudando Node, o conhecimento em MongoDB vai ser quase que obrigatório.

Escolha um Framework.

Webservices REST

Como eu falei, hoje em dia, pensou em Backend, já se pensa em criar uma API, nesse caso, é imprescindível que você tenha conhecimento sobre **Restful** e **Webservices** em geral.

Uma vez que você aprende, independente de ser em PHP ou Node, vai ver quão fácil é criar um webservice. Depois de criar 2 ou 3, você vai pegar o jeito da coisa, confia em mim.

Conhecimento mínimo sobre servidores

Outras tecnologias:

Além disso, existem algumas tecnologias essenciais para um bom programador Backend, que são:

WebSockets

GraphQL

ElasticSearch

São conceitos mais avançados, só que são a diferença entre um BOM programador e um EXCELENTE programador.

MOBILE

Ok, agora chegou a hora de vermos sobre o que precisamos aprender pra nos tornar um excelente programador Mobile.

Formas de programar Mobile

Hoje em dia, você pode criar aplicativos para basicamente Android e iOS. Sim, existem outros, mas irrelevantes atualmente.

São duas plataformas completamente distintas, com códigos distintos e formas de se fazer as coisas bem diferentes.

Existem hoje 3 formas de você programar Mobile:

Pra quem tá iniciando no mundo mobile, não recomendo o **100% nativo**, simplesmente porque a balança de coisas positivas não compensa o lado negativo.

100% nativo

Você vai estar usando a linguagem ORIGINAL de cada plataforma, no caso do Android, é Java/Kotlin, no caso do iOS, é Swift.

Vantagem: Controle total e absoluto, integração inédita, novidades totalmente disponíveis, ou seja, aquele update com novas funcionalidades que saiu ontem, já está disponível pra ti.

Desvantagem: São duas linguagens COMPLETAMENTE diferentes, ou seja, você vai ter que aprender a programar duas vezes, a não ser que queira se focar única e exclusivamente em UMA plataforma.

Um **aplicativo híbrido**, explicando a grosso modo, é um aplicativo que dentro dele, tem um navegador que abre do tamanho da tela e carrega um site.

Isso mesmo, você está lá manuseando o app achando que é um app, mas na verdade é apenas um site carregado dentro de um navegador que está dentro do seu app.

Eu recomendo o híbrido? Sim, por que não? É uma ótima opção que vai te possibilitar desenvolver cross-plataform, ou seja, um só código para múltiplas plataformas. Você só não vai ter toda a disponibilidade de funcionalidades que teria no 100% nativo.

Híbrido

Com o desenvolvimento híbrido de aplicativos, você consegue criar um único app e ele conseguirá funcionar tanto no Android quanto no IOS, isso é excepcional.

Vantagem: O custo de desenvolvimento é muito menor, você só precisa aprender uma gama de tecnologias.

Desvantagem: Perdas bem consideráveis em performance do app, bem como integrabilidade. Se sai uma novidade, você precisa esperar o pessoal da plataforma híbrida fazer as devidas implementações e traduções pra que você possa ter acesso àquela funcionalidade, quase sempre de forma limitada.

Claro, antes disso você precisa aprender **HTML**, **CSS** e **Javascript**, pois são componentes essenciais para a criação dos apps

ionic

O Ionic tem grandes vantagens como recentemente o **Ionic React**, em que você consegue usar código React dentro do seu app, o que já melhora bastante a performance.

Nativo-compatível

Com o nativo-compatível, você vai unir o melhor dos dois mundos, ou seja, você vai criar um único código, geralmente usando Javascript, que na hora de gerar o app final, esse código vai ser interpretado e transformado em código nativo, sem que você precisa criá-lo.

Vantagem: Além do fato de você só precisar aprender uma gama de tecnologias, com o nativo-compatível, você tem MUITO mais acesso às funcionalidades e manipulações diretas do dispositivo que um app híbrido.

Desvantagem: Você vai sempre ter que esperar alguém (ou você mesmo, por quê não?) fazer o processo de integração entre novas funcionalidades e a tecnologia que você está usando.

Existem hoje duas grandes tecnologias no mercado: **Flutter** e **React Native**.

Em termos de performanceo Flutter se sai melhor, pois a integração dele com o código nativo é mais leve e simplificada.

Também temos uma vantagem com relação à facilidade de criar interfaces bonitas. Não que o React Native não consiga, mas com o Flutter é um pouco mais fácil.

Uma grande desvantagem, na minha opinião, é a curva de aprendizado. Não que ele seja difícil, mas o Flutter usa a linguagem de programação Dart, ou seja, você ainda vai ter que aprender Dart pra poder brincar com Flutter. Enquanto com React Native, o bom e velho Javascript é o que você precisa para começar.

Flutter

React Native

Além da comunidade mais madura, o React Native usa Javascript, ou seja, usa uma tecnologia já bem conhecida e difundida, de fácil aprendizado e reutilização, pois um programador Frontend pode, em questão de minutos, começar a criar seus primeiros apps sem precisar de muito conhecimento, pois ele já sabe Javascript.

Para resumir...

Escolha **Flutter** se você quer uma tecnologia com excelente performance nativa, facilidade de desenvolver boas interfaces.

Escolhe **React Native** se você quer uma tecnologia que usa o já extremamente popular Javascript, bem como a ajuda de uma grande, forte, estabelecida e madura comunidade.

- Ambas foram feitas para multi-plataforma, ou seja, um código, várias plataformas.
- Ambas as tecnologias são originárias de grandes empresas, React Native veio do Facebook, Flutter veio do Google. Isso é positivo, pois não foram 2 caras sem compromisso que podem abandonar o projeto do nada. São grandes empresas que estão fazendo grandes apostas em suas respectivas tecnologias.
- Ambas são open-source, rápidas e gratuitas.
- Ambas tem documentações completamente detalhadas e atualizadas.
- Ambas vão dar ao usuário final uma experiência realmente nativa, sem travamentos ou limitações de interface.

Você precisa saber as linguagens usadas

Javascript (React Native) ou Dart (Flutter). Então começa por aí

Aprendenda todo o básico de cada tecnologia

Você vai aprendendo todo o básico de cada tecnologia, como a organização do código, onde colocar cada coisa, como colocar texto na tela, criar objetos, etc

Estude manipulações e eventos

A partir daí estude manipulações e eventos, ou seja, clique em botões, input de texto, seleção, animações básicas.

Transição de tela

Vamos partir para transição de tela, afinal, um app não é uma só tela, são várias.

Agora que você já consegue criar umas telinhas e mudar entre uma e outra, você já é capaz de criar uma gama boa de aplicativos, mas tá na hora de ir pro próximo nível, a integração com a internet.

Integração com a internet.

Aprenda como fazer requisições à webservices, pegar os dados, guardar os dados no app, exibir listas de dados, etc.

Agora que você já faz requisições, troca de telas e manipula elementos na tela, chegou a hora do gran master, ou seja, as integrações com o próprio dispositivo.

Integrações com o próprio dispositivo

O que quero dizer com isso? Câmera, notificações, vibrar, timer, teclado, sons. Ou seja, agora você vai aprender a usar o poder do próprio dispositivo, um luxo que só os apps nativos têm com excelência.

A partir disso, pronto, você já consegue criar qualquer app, o próximo passo é praticar, praticar, praticar.

FULLSTACK

Ah, quer ser FULLSTACK?
Primeiramente, você vai precisar de tudo que a gente falou acima, mas não entre em pânico, pois você precisa fazer uma coisa de cada vez

Você vai começar pelo Frontend

por um simples motivo, o ser humano gosta de ver progresso, principalmente no começo... E o Frontend vai te dar progressos visuais, então vai te dar a motivação necessária pra continuar, principalmente quando você tá começando e ainda está se decidindo se "isso é pra você mesmo"

Depois já pode pular pro Backend

Logo após os conhecimentos básicos de Frontend, você já pode pular pro Backend, que vai te abrir as portas pra possibilidades infinitas.

Quando você tem conhecimento de Frontend apenas, você fica um pouco limitado, o mesmo com Backend. Mas quando você adquire conhecimento nos dois e junta eles, nossa, seu cérebro vai explodir, é sério.

Está na hora de partir pro Mobile

Depois de aprender o básico do Frontend e do Backend, está na hora de partir pro Mobile, onde você vai utilizar muito do que você aprendeu nas duas outras áreas.