

UM PLANO DE ESTUDOS À
PROVA DE DESCULPAS,

GUIA DO DEV AUTO DIDATA



REATIVA TECNOLOGIA

*Agradecimento a todos os geniais programadores,
responsáveis pelas maravilhas tecnológicas do mundo.
Em especial ao [Larry Tesler](#), por ter criado o 'copy and paste'.*

Autor: [Paulo Luan Mariano Silva](#).

Co-autora e revisora: Ananda Gomes de Oliveira.

O guia do dev autodidata:

Proposta do manual: qual o motivo de ser tão barato?

Eu só consegui fazer esse manual a um preço EXTREMAMENTE baixo porque a maioria dos conteúdos NÃO são originais de minha autoria. Esse material seria gratuito, mas coloquei um preço bem baixo, para filtrar apenas as pessoas realmente interessadas em ler e seguir as dicas até o final. Todos os bons materiais já existem, e não acredito na necessidade de se recriar a roda. Se eu fosse criar originalmente todo o conteúdo que propus aqui, esse manual custaria MILHARES de reais e você chegaria no mesmo resultado.

Tomei essa decisão porque não acho justo que pessoas que já estão endividadadas, ou sem grana, ou que estejam perdidas, gastem dois mil reais para aprender algo que deveria estar à livre disposição no mercado. Este manual é um caminho organizado, filtrado e estruturado para você consumir vários cursos, materiais e palestras de diversas fontes disponíveis na internet, sendo a maioria gratuita.

O ouro desse material é o meu filtro pessoal e minha experiência para que você não perder tempo com materiais ruins e focar somente no que funcionou para mim. As decisões foram 100% sinceras, eu NÃO GANHEI NENHUM CENTAVO fazendo as indicações.

Não existe fórmula de bolo e nem receita secreta. O objetivo aqui não é te dar uma poção mágica e te vender uma mentira. Você **não está apenas** comprando um conteúdo, você está comprando a organização, a estrutura, o direcionamento, que vai te apresentar um **passo a passo validado, testado e acompanhado** para você **ganhar velocidade** para ser programador profissional.

Quem vai dizer ONDE e QUANDO você vai chegar é você mesmo!
Eu te darei a ponta, e você vai seguir na profundidade.

O plano a prova de desculpas para se tornar um programador profissional

Esse manual é o que eu escreveria para o Paulo Luan de 2009, que estava começando a carreira e completamente perdido sobre o que fazer para se tornar um programador “de verdade”. Eu tinha ambição, mas não tinha um “manual de voo” para conseguir chegar lá.

Se eu tivesse um manual, como este, eu teria evitado uma série de fracassos e perdas de dinheiro que tive ao longo da jornada e ***quero que você tenha a oportunidade de trilhar o caminho que te traga resultados reais, e que você seja uma versão muito melhor de mim.*** Obrigado por acreditar no meu trabalho. Vou fazer de tudo para merecer sua atenção e ser **objetivo e direto**.

Para fazer isso dar certo você terá que FOCAR. Acredite, cada dica tem um sentido prático. Neste arquivo vou evitar ser prolixo e ser **bem direto ao ponto das atividades práticas específicas** que você terá que seguir. **Evitei explicações sobre os termos e o racional por trás de cada decisão**, porque com essas informações mais densas o manual teria muita informação que você **não leria**. Você descobrirá estas informações pouco a pouco seguindo e estudando os materiais propostos por você mesmo.

O objetivo é te ajudar a **não se afogar em conteúdos**. O ouro do manual está em fazer você **focar! Não existem atalhos**, mas economizarei MUITO o seu tempo te mostrando o caminho **mais simples e rápido** de se tornar um **dev profissional** de **alto valor** para o mercado.

Eu demorei **11 longos anos** quebrando a cara para compilar esse material e selecionar **SOMENTE A NATA DO MELHOR CONTEÚDO** para você. Eu sei o que é ruim e o que é **perda de tempo**. Me dê o seu voto de **confiança e compromisso**.

Se você seguir tudo que eu falar aqui, **não tem outra**, você vai se tornar o programador que eu sempre sonhei em contratar.

Profissão democrática

T.I é uma das profissões mais democráticas que existem. Você literalmente só precisa de uma direção do que estudar (como este manual), um computador e internet. Você realmente não precisou de faculdade, de CREA, de regulamentadores, de NADA. Qualquer um pode ser programador, basta apenas gostar da profissão e ter aptidões e paciência de resolver problemas.

A natureza dessa profissão é a mudança, por isso NUNCA será tarde demais entrar no mundo da T.I, porém tenha em mente que você terá que estudar para sempre, porque as tecnologias sempre mudam, com exceção das bases (que é o que focaremos em aprender ao longo do manual).

Existem infinitas possibilidades e caminhos a se seguir em TI. Como já citei, minha proposta para você é seguir o caminho que você mais vai reaproveitar conhecimento e que no menor tempo o possível te dará acesso a oportunidades de mercado e bons retornos financeiros, condensados a partir da minha visão pessoal.

Bons estudos!

Passos iniciais:

🔊 *Background JOB - Do You Speak English?*

*Provavelmente você não verá inglês sendo pedido nas vagas de emprego, sabe por quê? **PORQUE É OBRIGATÓRIO!** A maioria dos conteúdos realmente bons de programação são em Inglês. Se você não souber até conseguirá se virar no começo, mas em pouco tempo já **não** vai sobreviver ao mercado. Como sou bonzinho, coloquei alguns recursos em PT-BR para você conseguir acompanhar, mas não fique mimado e foque em aprender inglês em paralelo.*

DICAS

1. Compre um caderno físico e tente ir anotando com as suas próprias palavras os conceitos e comandos que você aprender, e com o tempo você vai fixar melhor o conteúdo. Escreva como se fosse um tutorial para você mesmo.
2. Tente colocar tudo o que você for aprendendo no seu perfil do Github, de forma a ter muitos códigos, que vai ser uma prova inicial de que você é um programador esforçado (vai ser muito importante nos processos de entrevista).
3. Repare que indico várias fontes de conteúdos gratuitas, e quando pagas, serão a preços bem baixos, já que quero que a sua moeda de troca seja o seu **esforço** e **resiliência** e não a sua **condição social**.

Como fazer perguntas

Antes de qualquer coisa, a primeira coisa que eu queria ter contato quando eu era iniciante era o manual “*How to ask smart questions*” é um artigo que julgo OBRIGATÓRIO para qualquer programador, pois vai chegar um dia que você não irá encontrar as respostas das suas dúvidas na Google, e aí o que fazer?

Nesse caso você geralmente vai recorrer a fóruns como o Stack Overflow e Quora para perguntar para os devs mais experientes, mas... se você fizer a pergunta de maneira errada ou se a pergunta for boba você vai tomar patada. Por isso existem uma série de regras que você tem que entender para que você evite o sofrimento do coice de cavalo. Leia com atenção, esse conhecimento TRANSFORMA a sua vida e a sua postura na internet, e vai aumentar muito suas chances de ter a pergunta respondida sem sofrer com grosserias.

[Manual em Inglês](#)

[Manual em PT-BR](#)



PAUSA: TERMINE essa tarefa, de ler o manual, antes de continuar para as próximas.

Tendo as boas práticas em mente, é natural que ao longo do caminho você tenha dúvidas, recomendo que caso você pesquise em todos os fóruns, no Google, na documentação, visto exemplos e ralado muito e ainda assim não encontrou a resposta, então você pode usar os seguintes recursos para fazer perguntas: Grupos do Facebook ([ReactJS-BR](#), [Javascript-BR](#), [NodeJS-BR](#), StackOverflow [EN](#) e [BR](#), [Quora](#)) todos eles são beginner-friendly, mas sempre que fizer alguma pergunta siga todas as dicas do manual.

Antes de perguntar, leia a Documentação

Quando você travar ou tiver dúvidas o primeiro passo é olhar a documentação, existe uma ferramenta de ouro que reúne a documentação de todas as linguagens de programação, **ela deve ser a sua melhor amiga para o resto da sua vida**, que é o [DevDocs.io](#). Quando tiver alguma dúvida consulte-a antes de pesquisar por tutoriais de indianos escrevendo no bloco de notas em vídeos do Youtube. Entender os conceitos usando as documentações oficiais sempre será o melhor meio de encontrar informações confiáveis e completas.

Vamos começar!

De início você **não** precisa instalar nada no seu computador, tampouco mudar de sistema operacional, é possível você utilizar ambientes online para executar os trechos de códigos, sem precisar instalar nada na sua máquina. As plataformas que eu recomendo são:

<u>CodePen</u>	<u>CodeSandbox</u>	<u>JSFiddle</u>
--------------------------------	------------------------------------	---------------------------------

Elas são bastante completas e atualmente é possível rodar até códigos de Backend como NodeJS. Mas elas não substituem o ambiente da sua máquina, principalmente porque elas são limitadas e não oferecem a mesma experiência de programar localmente (no próprio computador).

HTML, CSS e JS

Qual o motivo de aprender HTML, CSS e JS?

Tecnologia é ferramenta e não time de futebol. O termômetro de qualquer escolha quando se trata de tecnologia será o próprio mercado. Seguir como desenvolvedor web te ajudará a ser um profissional de múltiplas oportunidades. Aprendendo essa base você vai estar apto a desenvolver sistemas em milhares de empresas ao redor do mundo e até abrir a sua própria empresa. Além disso, é o que mais dá poder de escolha: você poderá desenvolver sistemas web, desktop (ElectronJS), mobile (React Native), backend (NodeJS), tudo com a mesma base de conhecimento!

Na nossa primeira atividade, iremos utilizar a plataforma **FreeCodeCamp**, que já nos dá toda a infraestrutura para codificar os exemplos. Então vamos aos primeiros passos:

1. Crie uma conta no Github.

2. Faça o Login no FreeCodeCamp (usando sua conta do Github).

3. Inicie o módulo Responsive Web Design (ele é um passo a passo para você aprender o básico de HTML e CSS).

- *Atenção: Pare quando chegar no Responsive Web Design Projects.*

Se você for mais avançado, **ainda assim recomendo** você fazer, pois reforçará os conceitos que talvez você tenha deixado passar batido.



Seu esforço deve ser **diário** e **contínuo**. Defina um **horário** do seu dia e use sagradamente para finalizar esse objetivo, nem que seja somente 30 minutos do seu dia.

⚡ *Background JOB - Conteúdos adicionais, mas **não obrigatórios**.*

Outros recursos muito interessantes são:

[Dive Into HTML5](#) (livro extenso sobre HTML5 e suas features).

[Apostila da Caelum](#) que é um excelente e riquíssimo material.

[Curso da Udacity](#) (os conteúdos da Udacity são referências em qualidade, dificilmente haverá alguma escola de programação no mundo com tamanha qualidade).

🛑 **PAUSA:** Seu **foco diário** vai ser **apenas** na tarefa do FreeCodeCamp. Depois que você terminar essa atividade você **volta aqui**, que **continuaremos**.

TERMINE essa tarefa antes de continuar para as próximas.

Agora que você concluiu o FreeCodeCamp e conheceu os playgrounds, está na hora de começar a ver mais exemplos! Você tem muito mais capacidade de entender como

funcionam as estruturas da web, antes de começarmos a nova atividade teremos que falar de Linux.

Devo abandonar o Windows?

A maioria esmagadora das empresas e programadores usam Linux, mais especificamente Ubuntu. **Praticamente todos os programas e tutoriais de programação são feitos usando ele.** Se você continuar no Windows você não vai ter suporte, estará sozinho no mundo, conheço alguns devs que usam windows, mas eles se “viram” e configuram o ambiente sozinho (geralmente usando [Chocolatey](#)), eu particularmente te recomendo Linux (faça um dual-boot caso precise do windows para games por exemplo).

*Lembre-se que é **você** que tem que se adaptar ao mundo, e não o mundo a você.*

É praticamente inevitável migrar para o Ubuntu. Quanto antes você começar a usá-lo melhor. Você até consegue rodar os projetos web nos playgrounds que citei, mas não vai dar para continuar para sempre.

💡 Background JOB

Migrar para Ubuntu é um trabalho de segundo plano, não precisa ser feito agora, mas vai precisar ser feito em algum momento futuro - se quiser sofrer menos, faça isso agora.

Como vamos ter que aprender Linux uma hora outra, você terá que aprender o básico de quais são as motivações por trás desse sistema operacional e o racional do porquê utilizá-lo. A tabela a seguir contém as atividades relacionadas a aprender a utilizá-lo.

[Curso introdutório sobre Linux](#): Gustavo Guanabara é extremamente beginner-friendly e explica nesse curso sobre como é, para que serve e como fazer as configurações iniciais;

[Guia do Ubuntu para iniciantes do Fabio Akita](#): ele é muito objetivo e sincero, mas particularmente eu recomendo **MUITO** você assistir e ouvir o que ele tem a dizer. Nesse vídeo de 1h20m ele dá muitas dicas de como usar bem o Ubuntu, mas não se assuste se não entender algumas informações;

[Tutorial básico sobre a linha de comando](#): eu gosto bastante do canal Traversy média, tem um conteúdo bem direto ao ponto e é fácil de entender.

Terminou de assistir esses vídeos? Agora vamos treinar no [KataCoda](#). Lá você terá um ambiente seguro para rodar os comandos passo a passo e entender o que cada comando faz. Acredite, é muito importante você ter paciência para aprender Linux bem, já que você vai usar MUITO no dia a dia. Salve esse [link aqui](#) no seus favoritos do navegador, e sempre que você esquecer qual comando rodar, dê uma olhadinha lá. Esse [outro repositório](#) é extremamente útil no que tange Linux, merece ser outro favorito.

Aprendendo GIT

Depois do Linux, GIT vai ser a ferramenta que você mais usará **NA SUA VIDA INTEIRA!** 99.9% das empresas (e devs) usam. Ler o livro no [site oficial](#), é a melhor forma de entender os conceitos de como o GIT funciona, mas compreender os conceitos avançados do GIT é uma péssima alternativa **para quem nunca teve contato com ele**.

O que você tem que aprender agora é ser **pragmático**, entender como funciona o básico de mandar os seus códigos para o Github, focar no arroz com feijão ([como esse artigo por exemplo](#)), e depois vai evoluindo no entendimento das features avançadas do GIT.

Comece com:

[Iniciando no Git: Parte 1](#)


[Iniciando no GIT: Parte 2](#)

[Tudo que você queria saber, mas tem vergonha de perguntar](#)

[Documentação oficial](#) é um arquivo bem extenso, densa, e completo. Eu recomendaria você começar com algo mais pragmático, com outros tutoriais mais rápidos, mas se você realmente quer se diferenciar dos outros, leia a documentação completa e tente entender os conceitos por trás de cada coisa,

Infinitos links sobre Git.

Se você não gosta de ler, assista:

 Assista no [Youtube](#) aulas sobre GIT - Se ficar em dúvida comece a pesquisar conteúdos adicionais, saiba o que mais funciona para você. Dê preferência para conteúdos gringos como: [Traversy Media](#) ou o [Learn Git in 20 minutes](#), outros recursos legais são os vídeos do Akita, mas eles serão verdadeiros tapas na sua cara (se você nunca viu GIT ele vai te assustar; assista quando você já tiver usado GIT profissionalmente. Só o [vídeo 1](#) e [vídeo 2](#) eu recomendo você ver agora, porque é bom já ter a visão sincera, mas **por ora** não se importe de não entender os conceitos, deixe mais para o futuro).

Quando você tiver uma noção inicial de GIT, use o [KataCoda](#) para treinar GIT no terminal sem ter medo de dar um comando errado e explodir o seu computador. 😄

Quanto mais iniciante você for, mais perdido vai se sentir. TENHA CALMA!

Programação é abstrata mesmo, os conceitos vão vindo com o tempo, assim como você só aprende a dirigir dirigindo, certo? Então comece pequeno, entenda os comandos principais e vá se expondo aos pouquinhos nos conceitos mais profundos.

Tente escrever com as próprias palavras o que você entendeu do assunto. **Vá no seu tempo, descanse** e tente **re-assistir** ou **reler** o conteúdo de tempos em tempos.

ALERTA SALÁRIO: se você já fez **tudo** o que falei para você fazer, aqui você já pode pedir salários de **R\$1.500 a R\$2.000**. 💰💰💰💰

 **PAUSA: TERMINE** essa tarefa antes de continuar para as próximas.

CSS Profissional

Eu quero que você tenha um nível bom de CSS, a ponto de conseguir recriar interfaces de sites famosos como Netflix, Youtube, Instagram e etc. Para isto, quero que você faça:

Atividade

Fazer o [curso completo de CSS](#) do SoloLearn. Eles disponibilizam o aplicativo mobile e o site web, então use o que você se sentir mais confortável.



PAUSA: TERMINE essa tarefa antes de continuar para as próximas.

Depois de terminar, coloque o certificado do SoloLearn no seu LinkedIn. Agora sim vamos começar a entrar na parte profissional do CSS:

Atividade

Faça todos os Challenges do [FrontendMentor](#). Se conseguir, você está aprovado para o próximo nível!



PAUSA: TERMINE essa tarefa antes de continuar para as próximas.

ALERTA SALÁRIO: se você consegue clonar **PERFEITAMENTE** as interfaces de aplicações famosas com CSS, e **já fez tudo** o que falei para você fazer, nesse ponto você já pode pedir salários de **R\$2.500! R\$3.500!** 💰💰💰💰

Javascript

Agora chegamos na parte boa, a **programação**! Vamos começar com o básico do básico.

ATENÇÃO: Se você **nunca** na vida teve contato com nenhuma linguagem de programação, faça os seguintes passos:

1. Faça o curso de [lógica de programação](#) do Digital Innovation One (os cursos deles são meio chatos, mas se colocar 2x fica mais dinâmico 😊 tenha paciência, é gratuito e vai ser importante para o futuro);
2. Assista a [playlist de algoritmos](#) do Gustavo Guanabara;
3. [Python para zumbis](#) vai te dar uma introdução a algoritmos e pelo menos você terá um contato com uma linguagem de programação diferente (Python é legalzinho, mas o que vai te dar dinheiro é Javascript, por isso foque no JS - essa fase é só uma introdução);
4. Agora que você teve contato inicial com Python você vai conseguir acompanhar esse [curso introdutório](#) de 4 horinhas;
5. Se quiser investir um pouco mais em Python, o [curso do Renzo](#) é excelente.

Já sabe né? Termina isso antes de continuar.

Pronto agora você aprendeu o básico de algoritmos! Vamos seguir!

Atividade

Faça o Curso de Javascript básico do canal do YouTube [Curso em vídeo](#).

Agora que você já tem uma visão inicial do Javascript, vai ser mais fácil acompanhar o path de Javascript do FreeCodeCamp:

▼ JavaScript Algorithms and Data Structures Certification (300 hours)

▶ Basic JavaScript	○ 0/110
▶ ES6	○ 0/31
▶ Regular Expressions	○ 0/33
▶ Debugging	○ 0/12
▶ Basic Data Structures	○ 0/20
▶ Basic Algorithm Scripting	○ 0/16
▶ Object Oriented Programming	○ 0/26
▶ Functional Programming	○ 0/24
▶ Intermediate Algorithm Scripting	○ 0/21
▶ JavaScript Algorithms and Data Structures Projects	○ 0/5

⚠ ⚠ **ATENÇÃO:** Sua última atividade do ciclo será o [Cash Register](#)! Depois disso **NÃO** dê continuidade nos próximos módulos que é o Front End Libraries Certification. **NÃO** os inicie.

Atividade

Agora que você tem a visão inicial do JS, você vai conseguir acompanhar o passo a passo do curso [Javascript algorithms and data structures](#) da FreeCodeCamp.

Em **paralelo**, faça o path de [JS no SoloLearn](#).



PAUSA: TERMINE essa tarefa antes de continuar para as próximas.

Aqui você vai ganhar mais dois certificados, um do FreeCodeCamp e outro do SoloLearn. Coloque-os em seu LinkedIn.

Agora você já tem uma visão FODA de Javascript e de front no Geral. Parabéns, você já tem conhecimentos mais profundos que **80% dos programadores JS do mercado**. Agora, vamos para mais uma atividade de portfólio:

Atividade

Faça o desafio de criar [30 aplicações em 30 dias](#) do Wes Boss, e coloque os resultados em seu Github como portfólio.

 **PAUSA: TERMINE** essa tarefa antes de continuar para as próximas.

Agora você terá que aprender boas práticas de desenvolvimento, preste atenção: **CLEAN CODE é uma das coisas mais importantes quando se trabalha em grupo**. É isso que vai te fazer ser diferenciado em relação aos outros devs. Olhe e estude com atenção [esse repositório](#) e comece a aplicar essas técnicas em seus códigos.

 **PAUSA: TERMINE** essa tarefa antes de continuar para as próximas.

💡 *Background JOB - Os Frameworks mudam, as bases não!*

[Algoritmos & Estrutura de dados](#) e [design patterns](#) são coisas imutáveis, você sempre vai utilizar independente do contexto. Por isso recomendo que você estude esses tópicos clicando em cada um deles.

Se quiser treinar algoritmos, você pode utilizar o [HackerRank](#), [Codewars](#), [Exercism](#), [RealDev](#), entre [outros](#).

ALERTA SALÁRIO: se já fez **tudo** o que falei para você fazer, nesse ponto você já pode pedir salários de **R\$3.000 a R\$3.500**. 💰💰🎉🎉

Ponto de inflexão: aqui você já descobriu se gosta mais da parte **criativa** do trabalho ou mais de **programação**. Se for mais criativo, siga o caminho de CSS, Design, UX e UI. Se não, continue e aprenda ReactJS e NodeJS.

ReactJS

Qual o motivo de aprender ReactJS?

Lembra que falamos do racional por trás da decisão web? O Termômetro da decisão de aprender React é o **momento do mercado**. Essa é uma das tecnologias que mais se consolidaram no mercado web nos últimos anos. Todos os milhares de sistemas e aplicativos criados com essa tecnologia vão precisar de manutenção, e adivinha quem vai estar preparado para ocupar essa posição? Você mesmo.

React abre portas para as **melhores empresas do mercado**. Além de usar os mesmos conhecimentos de base do HTML, CSS e JS, você poderá criar aplicativos desktop e mobile com essa mesma tecnologia. Então foque em aprendê-la em profundidade! É o que faz mais sentido, tanto em termos de oportunidade de mercado, quanto de reaproveitar o conhecimento para lançar suas ideias no mercado (pois tenho certeza que em algum momento você vai começar a querer lançar ideias de aplicativos ou sistemas na internet).

Todo o caminho anterior foi para você adquirir o conhecimento de base, que é o que mais falta no mercado, agora sim você está preparado para usar frameworks.


ReactJS básico RocketSeat	PT-BR
ReactJS do FreeCodeCamp	EN





Agora que você sabe o básico, **deixe de comer um lanche do McDonalds** e compre esse curso:

[React - The Complete Guide \(incl Hooks, React Router, Redux\)](#) do Academind. É o melhor e mais completo que eu já vi sobre ReactJS. Eu recomendo que você invista 30 conto e compre esse curso, porque vale a pena. No dia que escrevo isto ele custa **R\$ 21,99**.

Em português existe o [ReactJS Ninja](#), do Fernando Daciuk, está **R\$ 145,00** atualmente, mas sempre aparecem alguns cupons de desconto.

 **PAUSA: TERMINE** essa tarefa antes de continuar para as próximas.

 **DICAS DE OURO:** Use [Storybook](#) para criar os componentes da sua aplicação, assim ficará extremamente fácil identificar se os componentes atendem as diferentes versões dos requisitos, e será muito mais fácil dividir e testar os componentes individualmente. Outra dica de ouro é o [bit](#), que é uma forma de compartilhar seus componentes entre aplicações e reaproveitar o máximo de código.

ALERTA SALÁRIO: se já fez *tudo* o que falei para você fazer, e já entende de *Redux*, *Sagas*, e Testes unitários com **JEST**, nesse ponto você já pode pedir salários de **R\$ 3.000 a R\$ 5.500**.    

NodeJS

Qual o motivo de aprender NodeJS?

Eu sempre bato na tecla de reaproveitar conhecimentos. Todas as aplicações precisam de um backend, que é a “parte de trás dos palcos”, onde a mágica por trás de todos os sistemas acontece. Lá é onde as informações sensíveis estão armazenadas e todos os processamentos pesados acontecem.

Aprender backend vai te dar uma noção de como tudo funciona de ponta a ponta. E não existe nenhum motivo no mundo para você usar outra tecnologia a não ser Javascript para fazer o backend, pois é o que você é mais especialista no momento. Fora que, você vai aprender qual é a posição que você mais se encaixa, se é no Front ou Back. Isso vai permitir você descobrir sua paixão de forma “barata”, aprendendo rápido e descobrindo quais são suas aptidões.

Quem disse que você precisa gastar 5 anos da vida e 50 mil reais para descobrir que você gosta mais de uma tecnologia ou outra? **Foque em errar barato.**

P.s: se for optar por algum banco de dados, o MongoDB é o mais indicado, pois, adivinhe qual é a sintaxe que ele utiliza? Isso mesmo! Javascript! Claro que isso não impede você de aprender outras tecnologias como o SQL, mas como o nosso foco é errar barato, MongoDB vai te permitir **ser mais produtivo em menos tempo**, e casar as tecnologias que você **já tem conhecimento**.

A qualidade mais importante de um DEV node é saber fazer testes unitários. Recomendo que você foque apenas em testes usando o JEST. Ele é o Test Runner, que vai dar para ser reaproveitado em toda a Stack, tanto no Front (ReactJS), Mobile (React Native) e Backend (NodeJS).

Comece tendo a visão geral da plataforma NodeJS com a apostila que está dentro da pasta 'bônus/NodeJS/1_apostila-iniciantes.pdf' (leia esse manual inteiro)

 **PAUSA: TERMINE** essa tarefa antes de continuar para as próximas.

O próximo passo é voltarmos ao FreeCodeCamp e agora fazer o [API's and Microservices Certification](#).

▼ APIs and Microservices Certification (300 hours)

- ▶ Managing Packages with Npm ○ 0/10
- ▶ Basic Node and Express ○ 0/12
- ▶ MongoDB and Mongoose ○ 0/12
- ▶ APIs and Microservices Projects ○ 0/5

 **PAUSA: TERMINE** essa tarefa antes de continuar para as próximas.

Agora você já tem uma visão intermediária, está na hora de se **aprofundar**.

Recomendo como curso: [NodeJS - The Complete Guide \(incl. MVC, REST APIs, GraphQL\)](#) da Academind. Preço: R\$ 21,99.


Agora você já é um DEV de conhecimentos avançados, comece a estudar um pouco mais sobre **arquiteturas de aplicações** e melhores práticas de organização de projetos. Para isto, leia o manual '**2_node_avancado**' na pasta '**bonus/NodeJS**', e comece a aplicar nos seus projetos da vida real!

 **PAUSA: TERMINE** é muito importante que você tente ler esse manual (imprima-o e faça anotações), mas termine! é um dos passos mais importantes!


Agora a dica mais importante de todas:

USE TESTES UNITÁRIOS PARA TUDO NA SUA VIDA.


Faça [esse curso](#) para entender o básico de testes usando chai, mas busque usar o Jest, que é o mais completo e o que faz mais sentido focar no momento.

 **PAUSA: TERMINE** a tarefa dos testes unitários antes de continuar para as próximas.


Assista [esta aula](#) de 1 horinha.


 **PAUSA: TERMINE** a tarefa dos testes unitários antes de continuar para as próximas.





Assista a playlist 'testes de software' [deste canal](#).

 **PAUSA: TERMINE** a tarefa dos testes unitários antes de continuar para as próximas.

Agora assista esta [excelente Playlist](#).

 **PAUSA: TERMINE** a tarefa dos testes unitários antes de continuar para as próximas.

 **DICAS DE OURO:** Comece a automatizar o seu código, use o [plop](#) para criar geradores de código para o seu projeto; tente sempre pensar em gerar código, vai ser a forma mais fácil de você começar a ser mais produtivo nos projetos. Principalmente em processos seletivos onde você precisa entregar **RÁPIDO** os desafios. Automatize CRUDS por exemplo.

ALERTA DE SALÁRIO: se já fez *tudo* o que falei para você fazer, e já entende de **React avançado**, **NodeJS avançado**, e testes unitários com **JEST avançado** (**mocks**, **stubs**, **instrospecção** e etc), boas práticas e Clean Code, e já fez mais de **10 mil commits** ou trabalhou na prática **8 mil horas**? Nesse ponto você já pode pedir salários acima de **R\$ 7.500**.    

Depois disso, se quiser ganhar mais, ou você terá que ir para as capitais trabalhar como PJ, ou trabalhar remoto para países gringos.

Ganhar 100 mil dólares por ano

Se **você fez tudo que falei aqui**, você é merecedor de todo o sucesso do mundo, e você está completamente preparado para trabalhar como Sênior no Brasil e no mundo. Repare que esse plano, se seguido diariamente, será concluído em um tempo menor do que uma faculdade, que demora em média de **3 a 4 mil horas para ser concluída**. Se esse mesmo tempo fosse dedicado a seguir esse plano, **você estaria apto a ganhar 100 mil dólares por ano**. Então bora tirar isso do papel?

Primeiro você tem que começar a olhar as vagas do Stackoverflow Jobs: [olhe aqui as vagas de React](#) e aqui as [vagas de NodeJS](#). Comece a reparar o que as vagas de 100k dólares pedem, faça uma auto análise do que está fraco em você em relação a esses requisitos e trace um plano de estudos em cima disto, e lembre-se de colocar em seu Github.

Outra opção é usar o X-Team e Toptal para pegar trabalhos remotos. Já adianto que não é fácil passar, mas é super possível, e já conheci várias pessoas que trabalham em ambos.

Se tem uma voz na sua cabeça falando que você NÃO VAI PASSAR e te sabotando, **faça mesmo assim a entrevista!** É até melhor, porque aí você não vai ter a cobrança de NECESSARIAMENTE passar. Entenda que o que é importante aqui é o caminho e não o destino final. Ter consciência e autocrítica do que você errou é o que vai fazer você ficar cada vez melhor nas próximas entrevistas.


Devore os seguintes manuais:

<i>Interview Handbook</i>	<i>Coding Interview University</i>
---	--

Eles vão te ajudar a ir bem nos processos gringos. Faça um resumo escrito a mão de cada um deles, e tente aplicar na vida real.

Teste os seus conhecimentos de “brincadeirinha”, usando os enunciados das entrevistas e torne o seu Github um verdadeiro portfólio foda de aplicações de testes reais, acessando a pasta ‘bônus’ que eu deixei junto com o manual, lá você encontra testes de Backend e de Frontend.

Quanto mais desses testes você fizer, mais preparado estará para qualquer processo de entrevista como dev.

 **DICAS DE OURO:** a melhor fase para se preparar para entrevistas é quando você **JÁ ESTÁ empregado. NÃO ABAIXE A GUARDA**, sempre fique pensando em passos maiores para a sua carreira, sempre deixe o seu Github e LinkedIn atualizados, e sempre se prepare para vagas gringas. Se você estiver preparado para elas, você vai estar preparado **PARA TUDO NA VIDA**.

Bônus 1: Como se formar em Harvard de graça

Você já deve ter visto que os cursos das melhores faculdades do mundo estão disponíveis online. Sei que 99.9999999999% das pessoas que estão lendo isso **não vão fazer** esses cursos (e eu te desafio), mas eles são abertos e quero te dar pelo menos a oportunidade de saber que eles existem.

O site [Class central](#) separa, cataloga e organiza os melhores cursos das melhores faculdades do mundo. A [OSSU](#) organiza todo o caminho para você se formar em ciências da computação de forma autodidata. Não vou me estender nesse assunto, mas eu seguiria pelo caminho da ClassCentral, e faria o [CS50 de Harvard](#), caso você tenha interesse.

Bônus 2: como encontrar e filtrar qualquer outro conteúdo relacionado a programação?

A partir daqui a sua jornada estará em suas mãos. Para você filtrar os melhores cursos e não ser enganado, use [esse site](#). A comunidade organiza e filtra os melhores cursos online relacionados a tecnologia. Pare de gastar seu dinheiro com cursos merdas de **R\$ 5.997,00** por aí na internet.

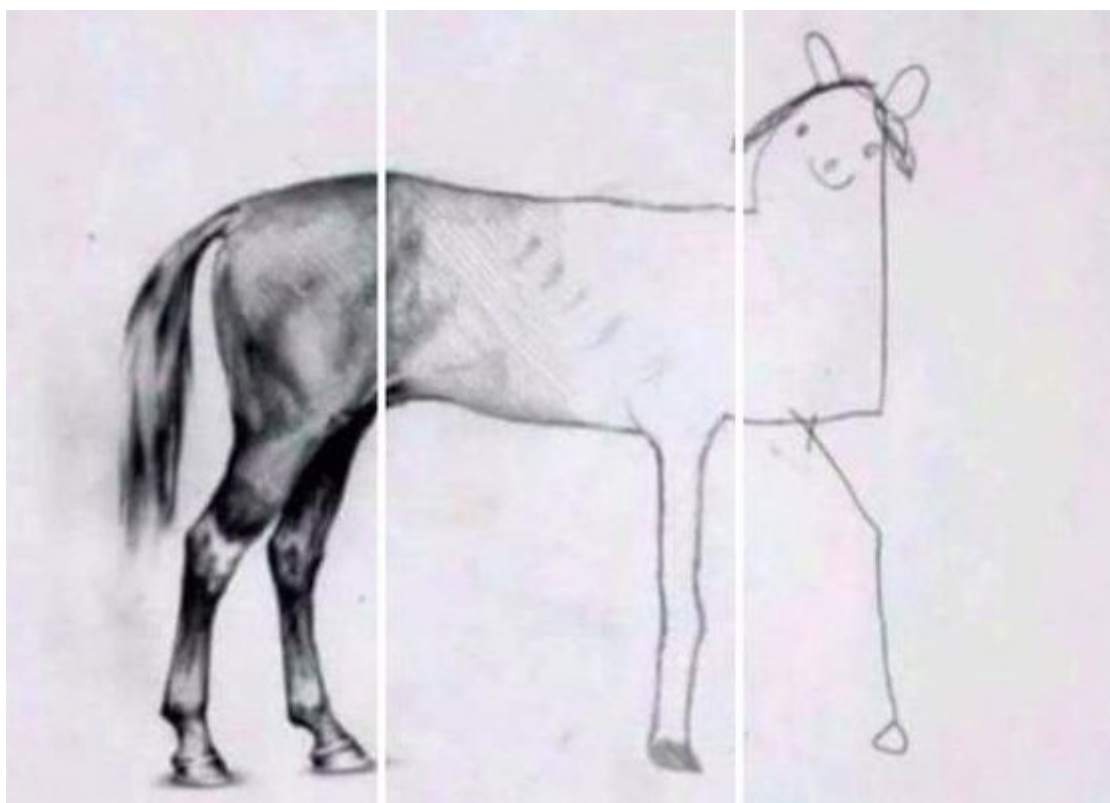
Palavras Finais

Por fim se eu fosse falar algo para o Paulo Luan do passado eu diria:

Não deixe “**a vida te levar**”, sempre **tenha um plano** de curto, médio e longo prazo.

Repare que esse roteiro de estudos não vai acontecer do dia para a noite, mas o que é legal é que é **100% possível**, só precisa do seu foco diário e de você mudar a sua mentalidade em **ter sempre um plano e segui-lo todo santo dia**.

Foque em **uma única coisa** por vez, **acorde mais cedo** e **programe o máximo o possível**, porque sempre vai ter um chinês ou indiano trabalhando mais duro do que você. A maioria das pessoas não vai ter foco para conseguir terminar as coisas que elas começam e **a vida delas serão o eterno ciclo de: começar animado, perder consistência, desanimar, fazer de qualquer jeito e depois largar mão**, como na imagem desse cavalo.



Sentimento não paga conta, então engole esse choro e **estude**. **Trabalhar duro e focado** é o que garantirá a sua vitória no longo prazo. Você vai se desanimar algumas vezes, mas o seu jogo é o **LONGO PRAZO**.

Consistência e direção é mais importante do que **velocidade e inteligência**.

O que importa é o foco diário, em 10 anos **ninguém jamais te alcançará**, e você não precisou se matar de trabalhar e estudar, porque o fez com inteligência e consistência ao longo do tempo.

Meu maior sonho era ter lido isso em 2009, e deixo aqui como um presente para você, que leu até aqui. Observe e conclua que esta é uma proposta que não é absurda, só exige coragem, organização e esforço.

Agora é com você.

Um grande abraço, [Paulo Luan](#) da [Reativa Tecnologia](#).