MARTINHO TOMÁS DALA

O DEV MODERNO

"O Inimigo do bug, é a gambiarra,
Seja um dev moderno, e fique longe de inimigos"

Martin Dala
LERMA EDITORA

“O DEV MODERNO”

O INIMIGO DO BUG, É A GAMBIARRA,
SEJA UM DEV MODERNO, E FIQUE LONGE DE INIMIGOS”

(Martin Dala)

Martinho Tomás Dala

LERMA EDITORA

© Martinho Tomás Dala & Lerma Editora. 2020.
Todos os direitos reservados e protegidos. É proibida a reprodução desta obra, mesmo parcial, por qualquer processo, sem prévia autorização do autor.

Editor: Martinho Tomás Dala
Colaboração e Revisão: Martins Gouveia
Abril/2020 Primeira edição
Lerma Editora Ltda.

Rua 10 de dezembro – Luanda, LA – Angola
Tel.: +244 924466025
E-mail: martindala40@gmail.com
Twitter: twitter.com/ martindala2
Facebook: facebook.com/martindala2

Github: github.com/martindala

Dedico este livro a todos programadores, desenvolvedores de angola e do mundo, que tenham amor no que fazem, e resolvam problemas das pessoas, porque essa é a nossa missão.

Prefácio

No princípio eu queria que um dos meus amigos programadores fossem autores deste prefácio, porque eu não tinha ideias do que ia pôr nessa parte do livro, mas ganhei coragem e decidi fazê-lo.

Desde o dia em que ganhei aquele computador de marca HASEE, minha vida mudou totalmente, eu não sei como me imaginar sem isso, graças a meu pai, hoje eu sou um profissional em T.I, porque quando recebi aquele computador, aprendi coisas básicas, minha rotina e estilo de vida mudou completamente, até que conheci programação.

Eu sempre quis escrever um livro técnico, na verdade vários, isto só para dizer que este não é o último livro que irás ler da minha autoria. Eu escrevia muito sobre ficção, roteiros para filmes, enredos e tudo mais. O primeiro livro que escrevi é sobre uma família de vampiros, talvez você nem goste muito dessa temática, mas eu escrevia mais livros sobre contos, romances sobrenaturais e coisa do tipo.

Mas eu sempre quis escrever um livro técnico da área que eu atuo, então surgiu esse Coronavírus que mudou a rotina de todos, todos em casa através do confinamento, decidi tirar essa ideia do papel em abril de 2020, e comecei a escrever, e só lancei em setembro por causa de vários contratempos que tive.

Então, está aí mais um e-book de programador para programadores, espero que gostes e que se identifiques com cada palavra desse pequeno e-book.

- Martinho Tomás Dala

Sumário:

AGRADECIMENTOS

Agradeço a Deus por ter me dado forças, disposição e motivação para escrever este e-book.

Agradeço muito aos meus pais, por eles sempre me apoiarem na minha carreira.

Meu maior agradecimento é a você, leitor, por interessar-se em aprender mais a respeito sobre programação e honrar-me com a leitura deste livro.

SOBRE O AUTOR

Olá aqui é o Martinho Tomás Dala, nas redes sou conhecido simplesmente por Martin Dala, programador, youtuber e um simples técnico informático nas horas vagas e como vês um amante da escrita.

Se você está aqui é porque você é um programador ou quer se tornar um e muito provavelmente, você quer se tornar um profissional em desenvolvimento e seguir todos atributos de um desenvolvedor moderno.

Eu sou programador desde os meus 13 anos, comecei talvez como você, ou não, na linguagem C. E sempre procuro se aprofundar e reter conhecimentos de livros e outros meios acadêmicos. Eu já vi um pouco de tudo, não posso dizer tudo, porque a programação é uma área muito vasta, e do que eu consegui ver, eu decidi escrever este pequeno e-book falando sobre o desenvolvedor moderno.

Toda experiência, estudos e pratica em programação me permitiram identificar pilares de um desenvolvedor moderno, isto porque o mercado de trabalho de hoje quer um desenvolvedor que consiga trazer soluções a todos seus problemas, por isso se você é um programador seja júnior, pleno, e sênior quero desejar as minhas boas vindas e que esse livro por mas que eu não tenha um doutorado em desenvolvimento de softwares, mas que você se identifique com tudo que você irá aprender e conhecer aqui e ajudar você a se profissionalizar nessa área que é muito top... Obrigado.

VOCÊ PODE ME ENCOTRAR EM:

- *Youtube*

Receba conteúdo em vídeo assim que for publicado ao se inscrever no meu canal do youtube:

Nome: Canal do Martin

- *Twitter*

Martin Dala (www.twitter.com/martindala2)

Sugestões, Criticas e Ideias, Enviar em: martindala40@gmail.com

INTRODUÇÃO

Este livro tem por objetivo fornecer aos profissionais envolvidos com o desenvolvimento de sistemas informáticos, os conceitos fundamentais e as técnicas de programação necessárias para desenvolvimento moderno de aplicações com novos padrões de criação de softwares.

Um software bem desenvolvido visa a uma arquitetura flexível que permita futuras alterações, facilite a produção de código organizado e legível, maximizando seu reaproveitamento.

Todo o paradigma da orientação a objetos, seus princípios e boas práticas procuram trazer esses benefícios para o software. E um desenvolvedor deve ser capaz de usar e conhecer todos esses paradigmas, boas práticas, padrões de desenvolvimento e arquitetura de software para se tornar um Desenvolvedor Moderno.

Todo software é desenvolvido com um propósito concreto, para resolver problemas reais que acontecem com pessoas reais. Saber resolver esses problemas seguindo padrões de desenvolvimento para que o software seja desenvolvido em perfeitas condições.

Por isso escrevo, este livro para mostrar, mas uma vez aos devs o que eles estão fazendo, sabendo que existem assuntos importantes para aprender e conhecer.

PARA QUEM FOI ESCRITO ESTE LIVRO?

Este livro é para qualquer tipo de dev, independente da hierarquia profissional, seja back-end ou front-end júnior ou sênior.

O QUE VOU ENCONTRAR?

Tudo que será abordado no presente livro é sobre algumas stacks e padrões que vou mostrar que farão de você um desenvolvedor melhor e mostrar que você é um desenvolvedor moderno. Encontrarás conceitos ou padrões muito conhecidos por desenvolvedores como MVC, POO, API entre outras feactures que foram criadas para facilitar o desenvolvimento de softwares.

Também vamos falar de frameworks, bibliotecas e plug-ins que por mais que existam pessoas que não gostem delas, elas surgiram com um único

objetivo: ajudar no processo de desenvolvimento. E elas se tornaram nas nossas melhores aliadas para desenvolvimento em qualquer plataforma, também vamos conhecer os famosos design patterns ou padrões de projeto, que todo mundo fala e alguns não sabem o que é, vamos tocar um pouco também na arquitetura de software, fundamentada por muitos cientistas da computação do mundo a fora, e vamos conhecer um pouco sobre segurança, e mostrar que se você é um dev e não se preocupa com segurança da sua aplicação você está distante de ser um dev moderno, e claro a sua credibilidade vai te deixar pra trás.

Agora, quero desejar boa leitura, porque se alguém que lê um livro nunca mais é a mesma pessoa, o dev que ler essas próximas páginas, nunca mais será o mesmo dev.

Obrigado.... Um abraço de angola.

CAPÍTULO 00

QUEM É O DESENVOLVEDOR MODERNO?

“O desenvolvedor moderno, é o profissional que inova nas formas de desenvolvimento em suas aplicações”.

Nesse momento você talvez estejas a se perguntar porque dev moderno? Quem é o dev moderno de quem ele tanto fala? **Desenvolvedor Moderno -** é o desenvolvedor capaz e consolidado na reinvenção de formas modernas de criar softwares.

Está meio confuso com a definição lá em cima, não se preocupe porque eu também estou, o dev moderno é um indivíduo, ou um profissional em desenvolvimento de software com capacidades de criar softwares modernos, isto é, com arquiteturas, infraestruturas, métodos modernos de desenvolvimento.

Eu acho que dessa vez foi menos confuso. Isso não quer dizer que devs de linguagens criada a muito tempo como Ruby, Pascal entre outras não são modernos, o dev moderno não é aquele que pega nas linguagens novas, eu não quero que você entenda desse jeito, mas sim que em vez de ser o indivíduo que usa linguagens novas para criar softwares, ele usa padrões de programação modernos e sofisticados que consistem em desenvolver um código limpo, um código que possa ser lido, um código sem complicações e um código com boas práticas de programação, e menos de gambiarra mesmo que as vezes seja a nossa única solução.

Porque existem muitos códigos sujos, digo sujo porque são mesmo sujos, um código que você depois de três anos, nem refatorar irás conseguir e nem lê. Então o bom programador é aquele que escreve um código pensando noutro programador e não somente no usuário, porque a maior das coisas que sempre pensamos é nos usuários, você não deve só pensar nele, mesmo que ele seja o chefe que põe o dinheiro no teu bolso, mais sim no programador mesmo que não seja um código open-source, se você trabalha em uma empresa você vai ter que aprender que o usuário nem sempre é o cara que devemos agradar. O que vai diferenciar você de um simples programador e de um programador moderno, é o conhecimento de deixar o software rodar com a mente descansada. Digo mesmo descansada porque se você trabalha em uma empresa grande por exemplo e você cria uma solução que pode comprometer a empresa com gambiarras e gambiarras, você não vai dormir com a mente sã.

Uma experiência que quero compartilhar sobre isso, é quando vendi o meu primeiro software, era um sistema para uma creche, fiz em CSharp(C#), com banco de dados SQL Server, nunca tinha vendido nenhum produto desde que conheci programação até aquele dia, fiz o orçamento e fui para casa todo motivado em fazer o sistema em três semana, era o prazo dado no orçamento, e não tinha que decepcionar o cliente, fiz o sistema mais o

problema não era fazer, porque eu sabia, o problema era implementar o sistema que fiz, e que estava funcionando no meu computador, mas, funcionar no computador do cliente, esse era o problema; meu computador era computador de um dev e não de um usuário comum, passei semanas para tentar instalar o sistema na máquina do cliente, por causa dos diversos erros que surgia, imagina um bug aparecer no momento de mostrar o software, esse é um pesadelo que não desejo a nenhum programador, mas infelizmente aconteceu comigo, o banco sempre dava errado, o id da tabela não tinha posto o incremento automático, por isso sempre pedia o id, um erro que tinha que resolver em frente do cliente, resolvi apareceu outros problemas e tudo mais, essa experiência é para mostrar que se naquela época eu conhecia de verdade tudo sobre padrões de programação, eu escreveria aquele código devidamente controlado, na época usei o C# procedural, onde dava erro na variável de ligação, e tinha que trocar em todos formulários, porque tinha várias variáveis de ligações, algo que você faz em uma classe hoje e bingo, mais foi um aprendizado que vou levar pra vida toda, então seja lá qual tipo de programador você é, web, desk ou mobile, todas dicas e padrões de programação que eu pesquisei e selecionei aqui serão muito úteis para você se você usar ao seu favor, e fizer do seu código um rei da selva que nem o leão.

E QUEM É O DEV NÃO MODERNO?

Antes de falarmos das feactures ou stacks que um desenvolvedor moderno deve ser profissional, devemos nos focar em aquele que não tem essas características para transformamos esse desenvolvedor em um dev moderno, onde o mercado de trabalho o siga onde ele estiver.

Eu poderia definir o desenvolvedor não moderno, como um desenvolvedor não autodidata, um desenvolvedor sedentário, sabe aquele indivíduo que não pratica exercícios físicos, é chamado por sedentário um indivíduo que sempre está acomodado em seu lugar de conforto, no contexto técnico é um desenvolvedor que não se atualiza ou não se atualizou com as novas tendências de se criar software.

- **O desenvolvedor não moderno** é o desenvolvedor de linguagem antiga?

Se você se faz essa pergunta é porque não entendeu bem o que de facto é um desenvolvedor moderno, qual é a linguagem que você considera a mais antiga, Ruby? Pascal? Assembly? Seja qualquer delas, elas não são de agora, então para mim, ser o desenvolvedor moderno tenho que deixar o Ruby ou pascal e migrar para nova tendência, algo como Javascript ou Python que é

muito rotulado como o futuro? A resposta é não, se eu te contasse as empresas e softwares muitos famosos no mundo dev que tem como Ruby entre outras linguagens consideradas antigas fizeram e fazem até hoje, você iria se surpreender, agora imagina se empresas grandes como NASA ou Microsoft usam as vezes essas linguagens, quer dizer que elas não são formadas por devs modernos? Ai é que você se engana, porque o dev moderno não tem nada haver com o tipo de linguagem que trabalha, mas se ele segue os padrões e feactures modernas de desenvolvimento, por exemplo a muito tempo atrás não existia o paradigma OOP ou Programação Orientação a Objeto, os programadores daquele tempo criavam seus softwares com o paradigma procedural mesmo, naquela época era necessário usar, mas hoje quase um terço dos programadores do mundo todos migraram para o POO, porque de facto tem vantagens que depois vamos falar aqui, mais eles se reinventaram, a acompanharam a evolução e são devs com objetivo de cada dia procurar métodos e soluções para resolver seus problemas de forma rápida, segura e muito controlada, imagina você um dev de hoje não saber POO, muito provavelmente você ficará para traz para aqueles que já tem esse conhecimento e estão juntos na candidatura de uma vaga de desenvolvedor.

Então o desenvolvedor moderno é o indivíduo autodidata, que por sua vez segue os novos paradigmas de desenvolvimento. Já imaginou o PHP por exemplo sem atualização que nem Java, sem feactures novas, sem métodos novos de desenvolvimento sempre no método arcaico, o mundo está mudando, e a tecnologias estão a implementar novas formas de se criar produtos tecnológicos, por isso se você se identifica que você não é um dev moderno, você será agora. E eu não quero fazer você desistir do paradigma procedural para o POO, é importante que você saiba os dois, porque nem sempre o POO resolve, as vezes o procedural leva menos tempo de desenvolvimento e mais estabilidade e leitura do código, por isso aqui também vou explicar que nem sempre aquela nova stack é a solução, devemos saber usar as tecnologias para que resolvam nossos problemas em conjunto de forma rápida, segura e controlada.

FEACTURES e STACKS? O que são?

Lembra o que eu disse, que o dev moderno não é aquele que usa linguagens da nova era? Sim, porque ele usa todos esses indivíduos para fazer do código um código legível, puro e fora de gambiarras, eu não consegui dar um nome para eles todos, mais vou chamá-los de Stacks

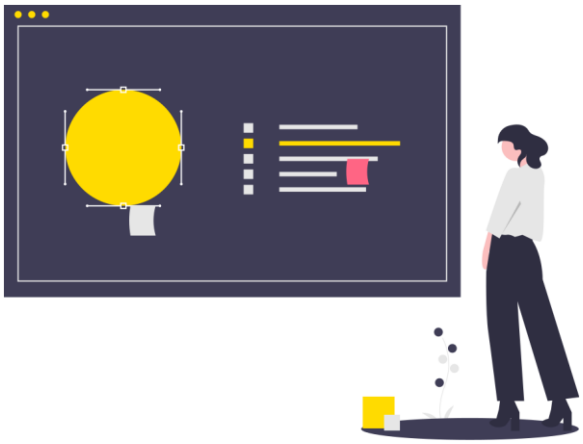
Mas o que são tudo isto se não são linguagens, vamos imaginar o seu computador com Windows XP em 2020, como você se sentiria, muito distante do mundo moderno né, pelo menos é como me sentira, não sei como vou exemplificar isso, mais elas são métodos de desenvolvimento, elas ajudam o desenvolvedor na criação de códigos mais sofisticados, limpos, puros, fácil leitura, fácil fatoração, fácil desenvolvimento, e boa forma de desenvolver softwares, sei que não foi a melhor definição do mundo, mas você viu a palavra fácil repetidas vezes nê?!, foi porque o fácil desenvolvimento é o que essas stacks fazem por nós. Para uma melhor compreensão vou definir cada uma delas para seu entendimento, então nossas próximas páginas você verá realmente quem são essas stacks que eu selecionei e os principais causadores de eu ter escrito este livro, as stacks que um desenvolvedor além de conhecer deve ser capaz de fazer.

Obrigado mais uma vez e boa leitura...

CAPÍTULO 01

PADRÕES DE PROJETO

“Padrões são consequências de aplicações com boas práticas ”.



Os padrões de projeto foram obtidos a partir das ideias apresentadas por Christopher Alexander em 1977, que sugeriu haver padrões comuns de projeto de prédios que eram inerentemente agradáveis e eficazes. O padrão é uma descrição do problema e da essência de sua solução, de modo que a solução possa ser reusada em diferentes contextos.

O padrão não é uma especificação detalhada. Em vez disso, você pode pensar nele como uma descrição de conhecimento e experiência, uma solução já aprovada para um problema comum.

"Padrões e Linguagens de Padrões são formas de descrever as melhores práticas, bons projetos e capturar a experiência de uma forma que torne possível a outros reusar essa experiência. "

Os quatro elementos essenciais dos padrões de projeto foram definidos pela 'Gangue dos Quatro', em seu livro de padrões, aqui segue a lista:

1. Um nome que seja uma referência significativa para o padrão.

2. Uma descrição da área de problema que explique quando o modelo pode ser aplicado.

3. A descrição da solução das partes da solução de projeto, seus relacionamentos e suas responsabilidades.

4. Uma declaração da consequência, os resultados e compromissos da aplicação do padrão. Pode ajudar os projetistas a entenderem quando um padrão pode ou não ser usados em uma situação particular.

Essa não é uma descrição do projeto concreto; é um modelo para uma solução de projeto que pode ser instanciado de diferentes maneiras. Uma solução comum para um problema em um determinado contexto.

Dizer que um padrão é uma solução comum implica que nenhum padrão é "criado", mas sim documentado. Um bom exemplo é o livro da Gangue dos Quatro (Gang of Four), Design Patterns: elements of reusable object-oriented. Para que exista uma solução, é preciso que primeiro se tenha o problema. Todos os padrões tentam resolver algum tipo específico de problema, e é assim que eles são classificados.

Voltando ao livro da Gangue dos Quatro, os padrões são categorizados como de:

• Criação – problemas que envolvem criar objetos;

•. Estruturais – problemas com a arquitetura da aplicação;

•. Comportamentais – problemas com o estado interno e o comportamento de objetos.

Hoje em dia se espera que um dev moderno conheça esses padrões de projeto, porque tem uma infinita importância nas soluções de problemas em desenvolvimento. Como por exemplo o padrão "**Singleton"**, aquele que diz que *não podemos ter duas ligações do banco no nosso projeto*, só esse padrão descarrega o nosso sistema, porque duas ligações comprometem muito o sistema.

O design parten ou padrões de projeto na verdade fazem com que voce construa aplicação sem repetições de código, de forma inteligente. Às vezes esses padrões de projetos são muito difíceis de se encarar, e muitos fogem, eu mesmo já fugi, mais tenta estudar um parten de cada vez, e faça de tudo para entender, depois de entender voce pode começar a estudar os outros, acho o Singleton mais fácil de se implementar de todos parten, então se voce está iniciando, comece por ele, depois pega outros parten como o factory.

CAPÍTULO 02

DESENVOLVIMENTO OPEN-SOURCE

“Se não existisse código livre, a evolução tecnológica não chegaria a esse nível”.

O desenvolvimento open-source(código-aberto) é uma abordagem de desenvolvimento de software em que o código-fonte é publicado e voluntários são convidados a participar no processo de desenvolvimento, suas raízes estão no Free Software Foundation (www.fsf.org), que defende que o **código-fonte não deve ser proprietário, mas deve estar sempre disponível para os usuários analisarem e modificarem como quiserem**.

Havia uma suposição de que o código poderia ser controlado e desenvolvido por um pequeno grupo central, em vez de ser desenvolvido por usuários do código.

Os softwares open-source estenderam essa ideia, usando a Internet para recrutar uma população muito maior de desenvolvedores voluntários. Muitos deles também são usuários do código. Pelo menos em princípio, qualquer contribuinte para um projeto open-source pode relatar e corrigir bugs e propor novas características e funcionalidade.

No entanto, na prática, sistemas open-source de sucesso ainda contam com um grupo de desenvolvedores que controlam as mudanças no software.

O produto open-source mais conhecido é, naturalmente, o sistema operacional Linux, amplamente usado como um sistema de servidor e, cada vez mais, como um ambiente de desktop. Outros importantes produtos open-source são o Java, o servidor Web Apache e o sistema de gerenciamento de banco de dados MySQL. Os principais competidores da indústria da computação, como a IBM e a Sun, apoiam o movimento open-source e baseiam seus produtos em softwares do tipo. Existem milhares de outros sistemas e componentes open-source menos conhecidos que também podem ser usados.

As aquisições de softwares open-sources costumam ser bastante barata ou gratuita, pois geralmente é possível baixar esses softwares sem custos. No entanto, se você precisar de documentação e suporte, você pode ter de pagar por isso, embora os custos sejam usualmente bastante baixos. O outro benefício-chave do uso de produtos open-source é que, sistemas open-sources maduros geralmente são muito confiáveis. A razão para isso é que eles têm uma grande população de usuários dispostos a corrigir os problemas em vez de os reportar ao desenvolvedor e esperar por novo release do sistema. Os bugs são descobertos e reparados mais rapidamente do que é possível, diferentes de softwares proprietários.

A pergunta que não se cala é porque um desenvolvedor moderno deve se preocupar com desenvolvimento open-source? Se o desenvolvedor participar em um produto open-source dando seu contributo, o desenvolvedor ganha conhecimentos que nem mesmo o seu professor saberia, pois com a contribuição de projetos open-sources, você acaba conhecendo como os outros programadores estão construindo seus softwares, que métodos eles usam, vás poder se reinventar ou mesmo descobrir erros que outros estão cometendo, e com muito aperfeiçoamento, você se torna um dev ninja. Participar em projetos open-sources ajuda o dev moderno a de facto compreender diferentes tipos de códigos, e várias feactures de desenvolvimento, além de saber trabalhar em equipe, porque praticamente trabalhar em uma empresa é desenvolver open-source, porque você terá que trabalhar com colegas que escrevem códigos diferente de você, e você terá que saber trabalhar em equipe.

Falando nisso, vou contar uma experiência que vivi há tempos.

Desde o meu início na programação eu nunca gostei de trabalhar em equipe, sabe eu gostava de fazer tudo sozinho, seja o back e o front, eu adorava, mas uma vez tinha que trabalhar em equipe e foi muito difícil, eu me frustrei e o outro programador se frustrou e decidimos parar o projeto, porque na verdade ambos não estávamos acostumados em trabalhar em equipe, mais se de facto se eu participasse de projetos open-sources antes, eu ficaria mais controlado nisso, e conseguiria trabalhar, porque, a não ser que queira ser um programador freelancer para vida toda, você terá que aprender a trabalhar com os outros devs, porque você um dia terá que trabalhar, então para começar a se ensaiar , busca projetos em repositórios de códigos, e participe contribuindo em códigos abertos, ou mesmo crie um projeto e deixa o código livre para outros contribuírem e você acompanha o desenvolvimento, assim conhecendo a forma como os outros devs escrevem códigos e se familiarizar com diversas formas de desenvolvimento, isso é que ajuda muito o desenvolvimento open-source.

CAPÍTULO 03

ORIENTAÇÃO AO OBJECTO

“Orientação a Objeto é programar com consciência pensando no cliente e no colega”.

O termo orientação a objetos não é mais uma novidade para ninguém. Todo curso de programação, inclusive os introdutórios, já falam sobre o assunto. Os alunos saem de lá sabendo o que são classes, a usar o mecanismo de herança e vejam exemplos de Carros, Pessoas e Casas para entender classes e métodos. O que ainda é novidade é ver todas essas coisas aplicadas em projetos do mundo real. Usar POO é muito mais difícil do que parece e, na prática, vemos código procedural disfarçado de orientado a objeto.

A diferença entre código procedural e orientado a objetos é bem simples. Em códigos procedurais, a implementação é o que importa. O desenvolvedor passa o tempo todo em escrever o melhor algoritmo para aquele problema, e isso é a parte mais importante para ele. Já em linguagens orientadas a objeto, a implementação também é fundamental, mas pensar no projeto de classes, em como elas se encaixam e como elas serão estendidas é o que importa. Pensar em um sistema orientado a objetos é, portanto, mais do que pensar em código; é desenhar cada peça de um quebra-cabeça e pensar em como todas elas se encaixarão juntas e eu não vou te ensinar aqui o que são classes, métodos, herança e polimorfismo, eu simplesmente vou dizer porque é importante o desenvolvedor moderno saber disso.

A Programação Orientação a Objeto ou simplesmente POO é um muito importante para um desenvolvedor, porque ajuda na criação de códigos legíveis, reusavél e fácil de ser mantido, é um paradigma universal, independente da linguagem, se você aprende POO em PHP, você aprendeu para quase todas linguagens, e isso é muito bom, já o procedural não é bem assim, cada linguagem possui uma sintaxe diferente da outra, por isso faz do procedural diferente, já o POO mesmo que uses a sintaxe da linguagem, como talvez o modo de declarar uma variável, o modo de consumir uma API, o modo de mostrar algo na tela, mas o POO é um paradigma universal que sempre existirá classes, métodos e tudo que você já conhece.

Muitas empresas, grandes e pequenas usam esse paradigma por este motivo, por ser universal e fica mais fácil o trabalho em equipe, e não só, porque além disso, o POO tem suas vantagens além daquelas citadas lá em cima, os sistemas orientados a objetos são mais fáceis de mudar do que os sistemas desenvolvidos em procedural. Os objetos incluem os dados e as operações para manipulá-los. Portanto, eles podem ser entendidos e modificados como entidades autônomas. Alterar a implementação de um objeto ou adicionar serviços não deve afetar outros objetos do sistema.

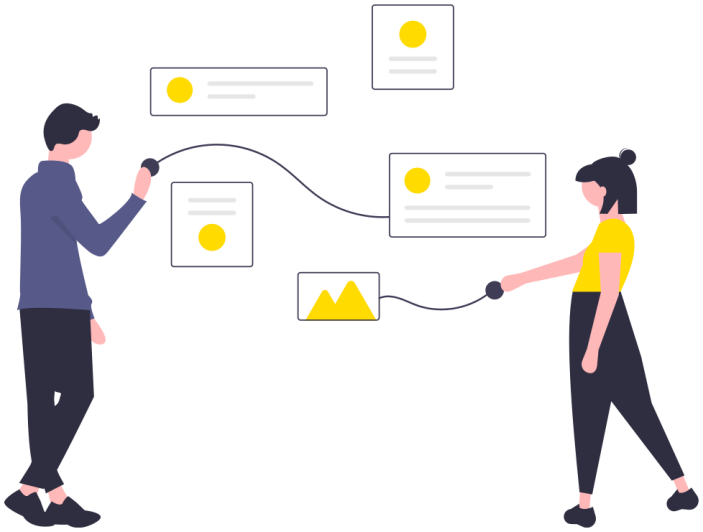
Mas isso não quer dizer que você deve desistir do procedural, os dois paradigmas de desenvolvimento são muito importantes e você programador moderno deve saber qual delas será necessário em cada problema que tiveres em desenvolvimento, porque imagina escrever um código que mostre todos números de um array em POO, seria muito código pra quase nada, nesse caso o procedural pode fazer isso em menos código; ou mesmo imagina você escrever um [Olá mundo] em POO, cria classe, método, e escreves "olá mundo", depois instancias, e pronto, todo esse código com cinco ou mais linhas de código em POO, que no procedural, você só precisaria de uma simples linha. Eu acho que você entendeu, nem sempre você vai precisar de usar o POO em tudo, é importante usar o POO em projetos que você sabe que será necessário e terá maior produtividade, e não usar só por usar, ou só porque o teu amigo usa, mas sim saber o porquê de usar.

Eu acho que ficou bem óbvio a importância não só do paradigma orientando a objeto como o procedural.

CAPÍTULO 04

WEBSERVICES (API, HTTP)

“O mundo se tornou mais interligado com os webservices”.



O que você pensa quando ouve a palavra webservice? Por que decidi pôr esse indivíduo como título desde pequeno capítulo, eu deveria só falar da API, mas nesse mundo de webservices tem muita coisa que o desenvolvedor moderno deve e precisa conhecer.

Antes de chegarmos a cereja do bolo que é a API, quero fazer uma pergunta, você sabe o que é HTTP? Ou melhor sabe fazer solicitações HTTP?

Bem, eu respondo, HTTP é o acrônimo de HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto), e é a base sobre a qual a web está construída. Toda transação HTTP consiste em uma solicitação e uma resposta. O protocolo HTTP por si só é composto de várias partes: a URL para a qual a solicitação foi direcionada, o verbo usado, outros cabeçalhos e códigos de status e, é claro, o corpo das respostas, que é o que geralmente vemos quando navegamos pela web usando um navegador. E eu acho que você já se identificou com isso, pensando nada mais nada menos em APIs.

- Então o que é API? E porque, eu dev moderno devo conhecer?

APIs (Application Programming Interface) ou simplesmente Interface de Programação de Aplicações costumam ser definidas como conjuntos de rotinas e padrões de uma aplicação que podem ser acessados por outra aplicação sem precisar conhecer os detalhes da implementação do software, ou seja APIs são simplesmente códigos feitos por outros devs que podem ser acessados na sua aplicação, é assim que gosto de definir, para ser mais compreensível, e eu acho que você entendeu. Em todo mundo, a comunidade de desenvolvedores é obcecada por APIs e isso vem crescendo, todo mundo quer consumir uma API, com ideias inovadoras que vem tendo.

Agora, com o desenvolvimento tecnológico de internet das Coisas e inteligência artificial, as APIs ganham um papel estratégico para o desenvolvimento de aplicativos em ambientes compartilhados. É muito difícil encontrar Apps Web ou mobile que não usem nenhuma API.
Elas também mudaram a face do desenvolvimento de aplicações. Ao expressar componentes de software em termos de suas operações, rotinas, entradas e saídas, padrões de programação, elas fornecem aos desenvolvedores blocos de construção com os quais podem contar para criar aplicações de forma mais rápida e eficiente.

O uso inicial geralmente é gratuito, o que torna o desenvolvimento e até mesmo os primeiros dias de lançamento das APIs muito mais simples. Mas, com o tempo, é preciso prestar muita atenção aos custos. Só hoje que falo para vocês, como já disse na nota, o mundo passa por uma peste conhecida por Coronavírus, e muitas entidades públicas criaram APIs para os devs poderem consumir e usar em suas aplicações. Então APIs é o futuro, e porque você deve aprender, eu acho que já respondi sua pergunta, quando disse é o futuro, se APIs são o futuro, então você precisa embarcar nessa.

O dev moderno deve saber consumir APIs, eu mesmo nem sabia consumir uma, mais eu notei a importância disso e é importante você saber consumir e criar, porque um dia terás que usar, e APIs ajuda os programadores, além de compartilhar informações, elas nos ajudam a criar softwares autônomos, e você sabe que o futuro é o autônomo, depois de ler este livro, de uma pesquisa sobre esse assunto, independente da linguagem que você programa hoje, você pode aprender a criar e a consumir uma api, imagina você criar uma aplicação para mostrar todas equipes de futebol do mundo, uma aplicação de notícias, você pode consumir uma API, que dispõe disso, e você só precisa lhe estruturar para apresentação.

Vou deixar um exercício para você. Entre nesses links, e consuma essas APIs, feita por um dev angolano, que é uma API com todos países de África, língua, PIB, código de telefone, capital, moeda, presidente, extensão, etc. Consuma essa API na linguagem que desejar e envia o screenshot no meu e-mail com a hashtag (#EuConsumiUmaApi). Estarei a tua espera.

Links:
Link do site da documentação:
countries-african.herokuapp.com

Link da API
countries-african-api.epizy.com

Link do GitHub do autor da API:
github.com/MartinsSilva

E-mail para enviar o screenshot:
martindala40@gmail.com

Webservices é um assunto muito longo para se estudar, não se baseia simplesmente em Apis, eu poderia definir webservices como conjunto de formas para invocarmos e utilizamos programas de terceiros em nossa aplicação.

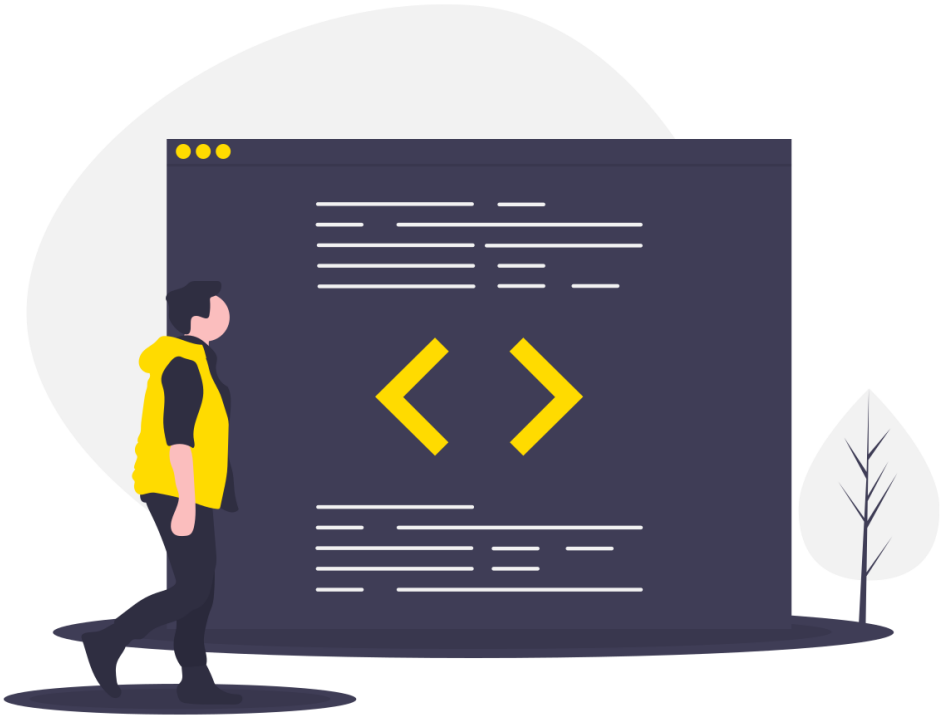
O objetivo dos Web Services é a comunicação de aplicações através da Internet. Existem muitas vantagens e benefícios para se aprender e suar webservices, uma delas é a *Integração de informação e sistemas*, uma vez que o funcionamento do Web service necessita apenas de tecnologia XML/JSON e protocolos HTTP, a comunicação entre sistemas e aplicações é bastante simplificada. Com um Web service é possível trocar informação entre dois sistemas, sem necessidade de recolher informação detalhada sobre o funcionamento de cada sistema. Os Web services permitem ligar qualquer tipo de sistema, independentemente das plataformas (Windows, Linux, entre outras) e linguagens de programação (Java, Perl, Python, etc.) utilizadas.

Outro benefício é a *Reutilização de código*, um Web service pode ser utilizado por várias plataformas com diferentes objetivos de negócio. O código do Web service é feito uma vez e pode ser utilizado vezes sem conta por diferentes aplicações.

Tem também a *Redução do tempo de desenvolvimento*, é mais rápido desenvolver com Web services, porque os sistemas não são totalmente construídos a partir do zero e facilmente são incluídas novas funcionalidades. E com maior *segurança*, o Web service evita que se comunique diretamente com a base de dados. Assim, a segurança do sistema que fornece os dados está salvaguardada. Existem muitos factores que fazem um dev moderno aprender hoje mesmo a usar os webservices.

CAPÍTULO 05

PADRÃO DE ARQUITECTURA MVC



MVC (***Model-View-Controller***) é um padrão arquitetural para organizar sua aplicação criado há muito tempo, e que hoje é bastante popular. A ideia por trás do MVC é dividir a aplicação em três grandes partes: a camada de modelo, onde vivem as classes e entidades responsáveis por todas as regras de negócio da aplicação, a camada de visualização, responsável pela interface com o usuário (em aplicações web, são os nossos arquivos HTML, CSS etc.); e a camada de controlador, que faz a ligação entre a interação do usuário na camada de visualização e as regras de negócio que vivem no modelo. Ter essas camadas nos ajuda a separar melhor cada uma das responsabilidades e, por consequência, ter um código mais limpo, reusavél e fácil de ser mantido.

Antes de aprender este padrão de arquitetura, a organização dos meus arquivos, me davam muita dor de cabeça, quando entrei no PHP, e não conhecia MVC, eu criava minha própria estrutura de organização, e meu estresse me consumia, imagina, em uma pasta 20 arquivos PHP, outros front, outros do banco, os erros eram constantes, mas pra falar sério conheci o MVC muito rapidamente e ai resolveu meus problemas, aquela organização de model, views e controller, foram muito legais, e até agora uso em tudo, e essa forma de organizar arquivos e os códigos da nossa aplicação são praticamente padrões incríveis, que eu acho que todo dev moderno deve conhecer. Porque além de acabar com seu estresse, deixa a estrutura de arquivos e códigos fáceis de ser encontrados e organizados para você como programador e para muitos que estarão trabalhando com você em equipe ou mesmo aqueles que contribuírem no código aberto.

CAPÍTULO 06

FRAMEWORKS, LIBS, PLUG-INS (AS AMIGAS DOS DEVS)

"Se você é iniciante, comece do início, deixa os frameworks para os veteranos".

Já conheci diversas pessoas nesse mundo, até aquelas que odeiam frameworks, libs e plug-ins. Porque alguém odiaria framework? Talvez seja sua pergunta, sabe, talvez seja porque eles não sabem o que realmente são frameworks.

Para falar sério eu também não gostava, e o meu odeio começou com bootstrap que hoje é o meu melhor amigo, o framework fronte-end do twitter, acho que você conhece, e muitos dos meus colegas também odeiam ele, mas depois de um tempo, comecei a usar, e percebi na verdade que frameworks são ferramentas que vem para minimizar o nosso processo de desenvolvimento, nos ajudar, então porque você odiaria alguém que quer te ajudar.

"São ferramentas criadas em linguagens de programação para amenizar o processo de desenvolvimento e criação de software"

Essa definição engloba todos eles, seja framework, libs (bibliotecas) e plug-ins. Existem diversos frameworks, e hoje eu considero Javascript como a linguagem com mais frameworks no mundo, entre os frameworks que você talvez conhece são: Bootstrap, AngularJS, VueJs, Laravel, Slim, Leaf, React.js, Django, entre outros. Libs como JQuery e plug-ins como OlwCarrousel. Tantas coisas que você pode pesquisar. O google dispõe de milésimos frameworks. Mas qual é a importância desses indivíduos para um dev moderno? Essa é a nossa pergunta chave, lembra?

O dev moderno, além de resolver problemas, ele resolve em tempo determinado, o dev moderno é aquele que quando o cliente falar quero esse sistema daqui a dois dias ou semanas, ele tem de trazer no mesmo dia, ou antes, esse deve ser o dev moderno, e como faço isso?, ai surge os frameworks que além de resolverem seus problemas em menos tempos, você tem mais produtividade, e muitos frameworks já vem englobado com técnicas e padrões universais de desenvolvimento, como orientação a objeto, rotas, padrão MVC, fácil consumação de APIs, fácil conexão com banco de dados, sabe tem framework para tudo, por isso os devs devem aprender e o tempo é precioso para o trabalho, por isso aprende um framework deixe de odiar frameworks, a não ser que quer trabalhar sozinho pra vida toda, você vai precisar aprender tecnologias que ajudaram você programador a resolver problemas com mais produtividade e profissionalismo, e vai mais uma coisinha, não quero dizer que você não precisa trabalhar com o puro, eu sempre digo, se você é iniciante, comece do início, nada de framework, mas

se você já é um veterano você saberá se de facto você precisara usar frameworks ou não, então é como eu disse quando falei do POO, nem tudo você precisara usar um framework também, mais pra um dev moderno, você precisará aprender e usar um framework, porque eles nos ajudam no desenvolvimento de nosso software.

Então caro inimigo dos frameworks ou comece a ver outro lado dos frameworks, ou não use eles, mas nunca odeie, é como os nerds; você critica eles hoje, mas um dia você terá que trabalhar para ele, porque hoje em dia a maior das empresas estão adotando conceitos de desenvolvimentos internacionais e com a demanda e tudo mais, você vai precisar de framework. Mas não use sempre, aprenda o puro também.

CAPÍTULO 07

SEGURANÇA

“Existe um hacker dentro de cada programador, e chegou a hora de acordá-lo”.

Sabias que você tem um hacker aí dentro, se você não sabia, saiba agora, e se você quer se tornar um dev moderno, você vai precisar acordar esse hacker por um tempo, porque o teu sistema não está seguro.

Viu o suspense (eu gosto de suspense), mais o importante mesmo é a segurança, já dizia um velho ditado, se você é programador e não se preocupa com a segurança, você está muito atrás e a tua credibilidade está muito comprometida, eu não quero que você seja o novo Edward Swonden ou o novo Kevin Mitinick, eu quero que você seja um programador preocupado com a segurança da sua aplicação, como um pai preocupado com a segurança do seu filho, seus sistemas devem estar seguros, não 100%, já que nenhum sistema é totalmente seguro, teu sistema deve estar seguro a 99%. Eu já vi programadores se preocupando muito com o layout do site, as cores, a funcionalidade, mais depois o podre do back-end, se vê logo de cara, eu não quero que você pare de se preocupar com design da sua aplicação, mas que outras coisas precisam de ser mais acompanhadas, a não ser que você é um web designer, tudo bem, mas se você se intitula o Fullstack Developer, o back-end Developer, você precisa se rever, porque se seu sistema está inseguro, você precisa estudar um pouco essa parte.

Sabe, ainda em pleno século XXI, tem programadores que criptografam os dados enviados do usuário com md5, eu tenho um amigo que odeia isso, ele sempre me lembra isso, se você não conhece md5 é uma função em PHP de criptografia muito barata e já existem decodes deles na internet, então você precisa investigar sobre possíveis ataques, como SQL Injection, CRF e vários outros que eu falo sempre no meu canal do youtube. Seja um autodidata, pesquise sobre segurança, leia livros, proteja seus softwares e a coisa mais preocupante quando, por exemplo você cria um sistema para uma empresa grande com contratos e tudo mais, e você mamela (gíria angolana para “despreocupado”) na segurança e compromete a empresa, meu caro, a cadeia te espera e a sua própria credibilidade fica comprometida, por isso, tenta se inteirar um pouco sobre segurança.

Lembra do que eu disse, acorde o hacker dentro de você, e todo programador tem um hacker dentro dele

Todo programador tem um hacker dentro dele porque um hacker também é programador, só que o hacker ele é mais ligado no estudo da segurança, então se dedique nessa área um pouco, e eu quero que você acorde o hacker dentro de você, para deixar de ver seu código com amor e carinho, porque se você olhar sua aplicação como um hacker você saberá quais são os possíveis ataques que seu sistema tem; e o programador em si, saberá

fechar isso, tenta hackear sua aplicação, faca testes para saber o quanto ela é segura e sua aplicação sairá dos 0% aos 99%.

Um dos meus sonhos como desenvolvedor, é se especializar em segurança da informação, não sei como hoje gosto disso, mais quando vejo uma informação sobre segurança, eu fico logo apaixonado, eu acho que a mídia foi um dos motivos, a ficção cientifica e tudo mais; e hoje como programador eu vivo na prática isso, e é muito bom quando você trabalha com algo que você ama.

Deixar sistemas seguros, trabalhar com back-end no tempo integral, são os pratos que todo indivíduo que queira ser o especialista em segurança deve comer.

CAPÍTULO 08

WEB VS DESK (PORQUÊ WEB?)

"A Terceira guerra mundial não é com governos e reinos, mas, sim com as plataformas**".**

A terceira guerra mundial das plataformas, qual das plataformas você gosta de programar, Web ou Desktop? Nessa terceira guerra mundial os lutadores são web versos desktop, esquece um pouco o mobile, porque todo mundo já sabe que ele vence os dois, isto consoante as pesquisas do mercado todo mundo usa um telefone, todos, e criar soluções que sejam possíveis ali, é uma coisa incrível para ao programador, mas para o dev moderno deve conhecer o mundo web.

Sei que existem programadores, que não gostam, eles amam o desktop, como amam seus amigos, mas porque web? E a resposta é simples, porque web é o futuro. Eu mesmo conheci a poucos meses uma garota que trabalha com banco de dados e migrou recentemente para mobile, e ela odeia web para caramba, não sei porque, se quer conhecer ela acompanhe o meu canal do Youtube tem um vídeo que fiz com ela.

Eu também gostava imenso do desktop, era a plataforma que eu mais programava, quando aprendi CSharp(c#), criava sistemas, e se você acompanhou esse livro desde o início, sabe que já vendi um sistema com c#, pelo menos aqui no meu pais, quase todo programador já criou um sistema de gestão em desk, aqui dá muito dinheiro, muitas empresas querem sistemas de gestão comercial, escolar e outros, e eu segui essa bolha, e eu gostei muito, desenvolver desktop é uma coisa muito fascinante, mas infelizmente essa coisa está perdendo com a web, a web está ganhar todos, a maioria das empresas estão investindo em aplicações web, em sistemas de gestão web, porque independentemente do local, com acesso à internet você consegue acessar, já o desktop, você vai precisar de um profissional em rede para interligar os softwares instalados em máquinas da empresa, mas web você só precisa de internet. Por isso eu digo que é o futuro, por ser capaz de um indivíduo conseguir ter acesso a dados de sua empresa, sua escola, em toda parte do mundo. Por isso muitos devs estão migrando para web. Eu não quero que você desista do desk, continue se quiser, mas dê uma chance a web, porque além do dinheiro que você perde por não saber ela, você ficará para trás por não migrar para uma tecnologia futurística.

Sem a internet, não existiria a web, por isso a base da web é a internet, no desenvolvimento desktop, nós desenvolvedores temos que criar formas de o nosso software seja compatível com diversos sistemas operacionais, como Linux, Windows e outros, mas com a web, você não precisa de se preocupar com isto, porque o acesso é simplesmente a partir de um navegador. Enquanto o desk você só acessa em um computador, a web você consegue

acessar em diversas plataformas, seja no computador, telefone ou mesmo uma tela, com o design responsivo, que vamos falar logo. Então a web é mesmo o futuro, você consegue perceber se realmente pensar.

Mais não quero aqui dizer que Desk vai morrer, por mim ela nunca vai morrer, existem várias aplicações que necessitam mesmo ser desk em vez de web, nesse caso posso interligar um pouco a experiência do usuário, imagina uma calculadora ser web? O Microsoft Office ser web, se bem que já tem, um reprodutor ser web? Um VSCode da vida ser web? Ou mesmo um GTA5 ou Vice City ser web?

Que coisa né, seria horrível, por mim, eles sendo desktop está perfeito, e esse é um dos motivos que por mim desktop nunca vai morrer. Existem aplicações que desk teria uma excelente experiência de usuário e web também.

Por isso não quero aqui dizer para parar de programar em desk e vir para a web, não, só quero que você saiba escolher as plataformas com o seu mercado de trabalho e com a necessidade que aparecer.

CAPÍTULO 09

CÓDIGO LIMPO

"Deixa de sujeira, codifique como um adulto".

Clean Code, ou simplesmente Código Limpo, senhor programador o quê isso te diz? Além de ser uma coisa que você precisa entender, você já se deparou com isso em um instante?

Muitos programadores famosos a nível mundial, tem dito um pouco sobre isto, Ron Jeffries, autor do Extreme Programming selecionou em prioridades as características de um código sem sujeira, que são:

- ✓ Efetue todos os testes
- ✓ Sem duplicação de código
- ✓ Minimiza o número de entidades como classes, métodos, funções e outros.

Também Michael Feathers, autor de Working Effecively with legacy code disse que um código limpo é um código que mostra que foi cuidado por alguém, alguém que calmamente o manteve organizado. Alguém que prestou atenção necessária aos detalhes, alguém que se importou.

Já Dave Thomas, o pai do eclipse, disse que o código limpo ele tem testes de unidade, nomes significativos, ele oferece apenas uma maneira e não de várias, possui poucas dependências, o código deve ser inteligível.

Grady Booch autor do livro Object Oriented Analysis and Design with Applications disse que um código limpo é simples e direto, ele é tão legível quanto uma prosa bem escrita, ele jamais torna confuso o objetivo do desenvolvedor.

Muitos programadores dizem muito sobre código limpo, até o criador da linguagem C++(Bjarne Stroustrup) que diz que o código limpo ao lê-lo deve ser como ouvir uma bela música numa estação de rádio, ou mesmo ver um design de carro magnífico, muitos falam sobre isso, e o Robert Martin lançou a muito tempo um livro com o nome “Clean Code”, que fala muito aprofundadamente sobre isso, mas nós podemos ver que todos programadores se identificam com a mesma coisa, a eficiência, o legível, o inteligível do código ,o código limpo deve ser o código que facilita outras pessoas a melhorarem, deve ser aquele que depois de um ano, você não vai ter medo de refatorá-lo, o código deve ser escrito de uma coisa inteligível aos seres humanos, e melhor que eu, você já deve ter notado a importância disso na vida de um dev moderno, ainda hoje, existem programadores que inserem nomes em suas variáveis que não tem nada a ver, como “Nome da sua amada,” o nome da variável tem de identificar porque ela foi criada, se for uma variável pra pegar nomes de um formulário por exemplo porque não

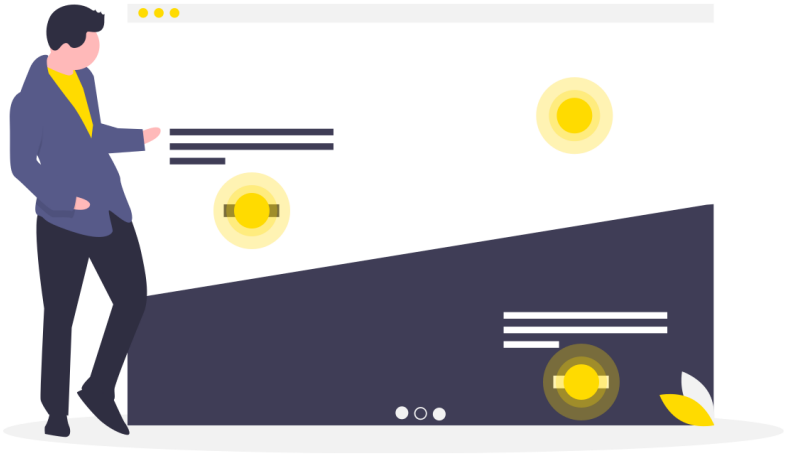
por “nome” mesmo, siga os padrões internacionais de escrever código e faça do seu código, um código limpo, sem muita gambiarra, faça testes, leia seu código, faça interpretação do código, leia como se estivesse a ler um livro de romance, deixe seu código legível e você vai notar as maravilhas que esse processo vai trouxer para você, porque se você é ainda aquele dev que escreve código como se estivessem a te dar corrida, você deve mudar agora.

Faça os programadores se apaixonarem pelo seu código.

CAPÍTULO 10

WEB SEMÂNTICA VS GAMBIARRA

"O inimigo de um desenvolvedor é a gambiarra, então fique desenvolvedor moderno e fique longe de inimigos**".**



Qual desenvolvedor não gosta de um belo café quente, com uma gambiarra ao lado?, eu próprio também amo esse lance, eu não sei quem inventou essa palavra gambiarra, se você sabe me fala no meu e-mail, deveria levar um Oscar, porque sei que todo desenvolvedor já um dia fez uma gambiarra, eu não quero que você desiste da gambiarra, as vezes é necessário, mas muitas vezes essas gambiarras comprometem muito a sintaxe da linguagem em causa e o seu padrão semântico, por isso muitos sites são poucos indexados em motores de busca como google, bing e outros.

O improviso muitas vezes toma conta dos programadores quando estão desenvolvendo seus softwares, criar uma lista com span em html. Criar um menu com section, criar títulos com a tag p, em vez de h1, isso tudo é improviso, outros não consigo me lembrar agora, mais existem vários, e você sabe disso, se você é um expert da internet, já ouviu falar de SEO (Search Engine Optimization), então, se você ir pesquisando regras de indexação do motor de busca do Google, você vai ter que melhorar seu código, porque os algoritmos de muitos motores de busca, seguem o padrão da web semântica, porque são a recomendações das linguagens, agora quando você tenta ser o vilão do filme, você terá problema em indexar seu site em motores de busca. Ter um código de página bem organizado e identificado irá ajudar muito nisso para sua página ser encontrada em motores de busca, o código HTML do seu site é fundamental para o SEO e é preciso saber trabalhar corretamente com as tags h1, h2, h3, img, títulos, meta dados, links e muito mais. Trabalhe corretamente o HTML do seu site seguindo as regras de semântica, isso você encontra no W3School, estude-as e compreenda a web semântica, porque hoje em dia as empresas estão preocupadas com sua posição nos sites de buscas, elas querem que sua marca, produto ou serviço sejam vistos e acessados por essa grande parcela de internautas então você como desenvolvedor moderno sabendo disso, para que o site dos seus cliente seja bem posicionado em motores de busca e tenha maior visibilidade e credibilidade; você precisa apostar na escrita de seus códigos.

PESQUISE SOBRE: SEO

CAPÍTULO 11

SEO

"Vê se cuida mais do teu HTML, porque sem ele o teu site não existiria".

Então, SEO, eu sempre que ouvia esse nome em livros, vídeos no Youtube, eu fugia, então aconselho a não fugir, é uma coisa grande para se falar, mais eu não estou aqui para ensinar você a usá-lo, mais para vermos os erros e a importância do SEO para um desenvolvedor moderno.

Search Engine Optimization, vou definir isso de uma forma bem simples, se você traduzi-lo do inglês seria “Otimização de Motor de Busca”, qual é o motor de busca mais famoso que conhece? Google nê, então eu sei que nessa caminhada de desenvolvedor, já ouviu muito aquele: meu site está a aparecer no topo do Google. E sabes porque aparece aí, por causa de você desenvolvedor, você tem o poder de pôr seu site ou do seu cliente no topo, só basta estudar e entender o que é SEO.

Mas simplesmente você pensa e pergunta Martin, porque eu deveria me preocupar em aparecer nos resultados de busca? Eu nem vou precisar te responder você vai se auto responder. Imagina você na barra de pesquisa do Google e pesquisar por “Angola”, por exemplo, e você não encontra nada, como você vai saber que existe algum pais com esse nome ou empresa outra coisa do tipo? Agora uma parte que dá dinheiro, imagina você cria um site para um cliente, hospeda e o cliente pesquisa no Google e não aparece o site dele, o que você criou, como ele se sentiria? Não seria conhecido, seria que nem um fantasma, a empresa perderia credibilidade, porque hoje em dia uma empresa que não tem um site, está muito atrás das que têm. Então será que tirei suas dúvidas?

O SEO é muito importante para quem tem um site até mesmo um aplicativo por exemplo no playstore, uma das formas de conseguir clientes e audiência é através dos resultados de busca, mas aparecer no topo do motor de busca não é a única vantagem do SEO, só citei essa porque é a mais comum, existem várias, mais vamos lá para parte técnica, você disse que o SEO depende de mim dev, como assim?

Acho que você já teve uma ideia em relação a isso quando falamos das gambiarras, por exemplo na web, já que estamos a falar de sites, o html é o indivíduo que muitas das vezes não tem ganhando atenção dos devs, existem vários frameworks para aprender, várias linguagens, e só usam o html porque é necessário e ignoram muitas vezes, mais esse mesmo individuo é o responsável por 90% do rankeamento do seu site em motores de busca só pra não falar 100%, ele o pai da web, e se você não se preocupa inteiramente

com ele, não o usa direito, faz várias gambiarras na sua sintaxe, usa tags que não deveriam ser usadas para determinados fins, ai você terá que se preocupar no futuro próximo, então recomendo a começar a dar um olho diferente no HTML.

Uma vez, eu estava conversando com um amigo a respeito disso, ele tinha posto o sistema que ele desenvolveu em produção, ele me explicou várias coisas relacionadas a SEO, até convidei para o meu canal do Youtube para falarmos um pouco sobre isso, e eu percebe que o SEO, é uma coisa muito importante que os devs devem se preocupar, todos nós estamos mais preocupados com outras coisas, desenvolvendo funcionalidades incríveis e esquecemos isso e as vezes ficamos sem conhecimento sobre isso, ele me explicou inteiramente tudo um pouco, falou que o Google tem um robô, e esse robô ele tem regras que segue para ranquear o seu site e esse robô é muito rigoroso porque ele vela nas coisas que passa muito na cabeça dos devs, como aquela imagem de 1GB que você mete no site porque é linda, as fontes de texto, os vídeos, e a semântica do HTML, eu percebi que o robô do Google ele não lê o teu CSS, nem JS, nem nada disso, ele lê sua página só com a informação, o HTML. Por isso o HTML é o órgão importante quando se está a estudar SEO, para teu site ser veloz, o Google dispõe de várias ferramentas para ajudar seu site ser o **Flash** da www, ter urls amigáveis e saber usar os keywords. Depois dessa leitura você precisa pesquisar mais a respeito de SEO, porque talvez o seu site precise de melhorias.

CAPITULO 12

GIT VS GITHUB

"Que tal deixar seu código no GitHub, e eu dou um star**".**

O porão do programador (Github) e o controle de versão (Git)

Eu mesmo já confundi os dois, é como confundir Java e o Javascript, mas essa nunca confundi, mas Git e Github, eu já, talvez porque os dois se integram quando usamos. E se você está na dúvida, vou tirar isso para você e depois iremos concluir porque um desenvolvedor moderno deve conhecer esses dois caras.

O trabalho em equipe é um grande desafio, nosso código tem que se integrar com o código de todos os outros membros da nossa equipe.

- ✓ Como podemos detectar que estamos alterando o mesmo código que um colega?
- ✓ Como mesclar as alterações que fazemos com a demais alterações da equipe?
- ✓ E como identificar conflitos entre essas alterações?

Fazer isso manualmente, com cadernetas ou planilhas e muita conversa, parece trabalhoso demais e bastante suscetível a erros e esquecimentos. Seria bom que tivéssemos um robô de integração de código, que fizesse todo esse trabalho automaticamente, ou seja um robô que integrasse ou juntasse o meu código feito no meu computador com o do meu colega de equipe feito no computador dele.

Existem ferramentas que funcionam como *máquinas do tempo* e *robôs de integração* para o seu código. Elas nos permitem acompanhar as alterações desde as versões mais antigas. Além de identificar conflitos, tudo de maneira automática. Essas ferramentas são chamadas de sistemas de controle de versão.

Nesse tipo de ferramenta, há um repositório, ou seja, um sistema de hospedagem que nos permite obter qualquer versão já existente do código. Sempre que quisermos controlar as versões de algum arquivo, temos que informar que queremos rastreá-lo no repositório. A cada mudança que desejamos efetivar, devemos armazenar as alterações nesse repositório.

Eu acho que você já entendeu, o sistema de controle de versão que é o famoso Git, e o repositório que guarda todas as versões do nosso código se chama GitHub.

O **Git** é um sistema de controle de versão que, pela sua estrutura interna, é uma máquina do tempo extremamente rápida e é um robô de integração bem competente e foi criado em 2005 por Linus Torvalds, o mesmo criador do Linux.

Depois de três anos, isto em 2008, foi criado o GitHub, uma aplicação web que possibilita a hospedagem de repositórios Git, além de servir como uma rede social para programadores. Onde você pode encontrar diversos projetos de código aberto importantes como jQuery, Node.js, Ruby On Rails, Jenkins, Spring, JUnit e muitos outros.

Você entendeu? ...

Imagina o seguinte, eu e você trabalhamos juntos em uma empresa, estamos a criar um sistema, eu com meu computador, e você com o seu também, mas estamos criando o mesmo sistema, para de facto a gente trabalhar junto, eu preciso integrar as alterações e funcionalidades que você vai pôr no sistema e você precisa também das minhas funcionalidades, isto para aplicação ser só uma, então com o Git, a cada funcionalidade que eu implemento no meu computador envio no repositório GitHub, e você recebe as alterações e integra no seu sistema partir com o git, o mesmo acontece quando você implementa algo novo, eu recebo a partir do repositório da nossa equipe. Eu acho que com essa explicação, você entendeu muito bem. Agora porque é importante você desenvolvedor moderno conhecer isso?

Uma pergunta muito redundante depois da explicação lá em cima, eu acho, primeira coisa o trabalho em equipe, você está notando que esse tem sido a maior vantagem em aprender diversas tecnologias, o trabalho em equipe. O trabalho home office funciona do mesmo jeito, você vai precisar usar essas stacks, um desenvolvedor moderno deve aprender essas duas stacks que acho muito bom, depois daqui, pesquise mais sobre essas stacks e aprenda-os, vai servir muito para sua carreira como um profissional.

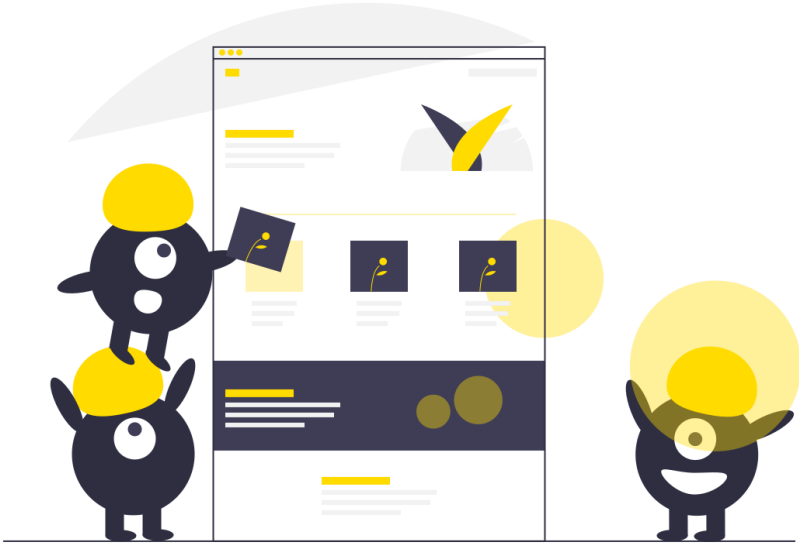
Eu comecei a trabalhar com Git e GitHub a tempo integral, porcausa da empresa onde trabalho, lá todos desenvolvedores trabalham em conjunto e a empresa tem uma conta no GitHub onde cada atualização, podem subir no servidor do GitHub, depois para pôr em produção. É fantástico quando você trabalha com essas ferramentas porque você começa a crescer de um simples desenvolvedor para o desenvolvedor moderno com D maiúscula.

Se quiser, pode me seguir lá no GitHub, [github.com/martindala], lá posto meus projetos pessoais.

CAPÍTULO 13

UI/UX DESIGN

"Capricha no UX da sua aplicação, e deixa de complicar o usuário**".**



Eu defino experiência do usuário (ux design) como a **forma** ou **processo** de **criar** experiência de usuário em um determinado **produto,** nesse caso o nosso produto é o software que desenvolveremos.

No decorrer do dia nos tornamos "usuários" de uma porção de coisas. O alarme do telefone que nos acorda de manhã, o controle remoto do ar condicionado, o Facebook que ninguém consegue viver sem ele, o multicaixa para fazer transações financeiras, a cadeira para descansar.

Quando você usa algum desses objetos, você tem uma experiência. Se você já passou nervoso na frente do multicaixa porque ele não entregou o dinheiro que você estava esperando e não deu nenhuma explicação sensata sobre o motivo da recusa, você possivelmente teve uma péssima experiência enquanto usuário do multicaixa. É esse tipo de experiência de usuário da sua aplicação vai ter? Uma experiência boa ou uma ruim igual à do multicaixa?

UX se trata sobre definir o problema que precisa ser resolvido (o porquê), definir para quem esse problema precisa ser resolvido (o quem), e definir o caminho que deve ser percorrido para resolvê-lo (o como).

UX designer é definir como as pessoas irão interagir com o produto, quais tarefas conseguirão realizar dentro dele, as características do UX são:

Usabilidade: garantir que as interfaces sejam fáceis de usar. O usuário consegue realizar uma tarefa sem transtorno ou demora?

Design de interação: entender e definir o comportamento das interfaces quando o usuário interage com elas.

Taxonomia: organizar e rotular a informação de forma que faça sentido para o usuário.

Estratégia de design: é o entendimento e definição dos porquês do produto. Para quem ele foi criado? Como o produto evolui com o passar do tempo?

Esse livro não é sobre UX, mas é algo que você deve se preocupar quando constrói seu software, para que depois não se perguntes porque minha aplicação não tem usuários, tem de dar uma picadinha de experiência de usuário, porquê o botão é cor verde? Porquê por carrossel? E tudo sobre isto. Entender ux, é uma tarefa importante para você que deseja ser um dev moderno, conhecer um pouco do seu público alvo.

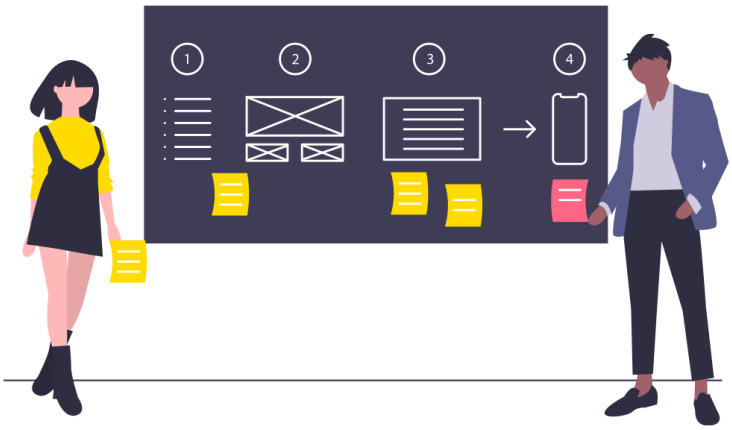
Hoje em dia tem de tudo no Youtube, até videoaulas de como criar uma conta por exemplo no Facebook, mais será que Facebook é assim tão difícil que precisasse de videoaulas para aprender a criar conta?, Tenho certeza que não? Mais existem sites e programas que para você instalar ou usar, você precisa dar uma googleada ou ver um vídeo no Youtube, então você como desenvolvedor deve pensar nisso tudo, uma das coisas que nos desenvolvedores não temos pensado é sobre isso, nós queremos é desenvolvedor as funcionalidades, por um carrinho de compras complexo da vida, ou um modal que fecha depois de dois minutos, para você está totalmente bom e perfeito, mais e para o usuário.

Sendo sincero, eu quando entro em um site e vem aquela modal perguntando se estou a gostar, e me dizendo para se-inscreva, eu fico muito aborrecido, que se não for por forças maiores que lá estou, eu saio do site imediatamente, os campos de texto, uma das coisas que aconselho muito sobre isso, os formulários de cadastro e login em aplicações, é sobre as APIs das redes sócias, é muito bom isso, Google,Facebook,Twitter já criaram e disponibilizaram gratuitamente suas apis para você usar em suas aplicações, e quase um terço dos internautas sabem que quando entrarem no site, deve encontrar a opção de entrar com Facebook por exemplo, porque eles não querem usar muito o teclado porque eles são preguiçosos, então você deve começar a pensar como usuário e entende-los.

CAPÍTULO 14

METODOLOGIAS ÁGEIS

“Seja um rei em metódologias ágeis, e deixe seu cliente feliz”.



Em fevereiro de 2001 um grupo de profissionais extraordinários do desenvolvimento de software reuniu-se em Utah para discutir melhores maneiras de desenvolver softwares. Esse encontro deu origem ao manifesto ágil, uma declaração com os princípios que regem o desenvolvimento ágil, o manifesto ágil também é composto por alguns princípios:

• Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face. Software funcionando é a medida primária de progresso.

• Contínua atenção à excelência técnica e bom design aumenta a agilidade.

Simplicidade - a arte de maximizar a quantidade de trabalho não realizado essencial

As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis.

Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Scrum, Extreme Programming (XP), são exemplos de métodos ágeis. Cada um deles traz uma abordagem diferente que inclui diversos valores, práticas e reuniões. Com esses métodos ágeis, você terá a capacidade de desenvolver seus softwares em menos tempo orçamentado pelo cliente, e com mais produtividade, queria falar um pouco sobre o Extreme Programming, é uma metodologia muito boa, por isso, nas referências vou deixar um número simbólico de livros para você encontrar esses livros, que tem sido fonte de pesquisa para mim e para muitos devs modernos.

As metodologias ágeis são métodos de gestão ideais para projetos que exigem rapidez, flexibilidade e foco na melhoria do desempenho da equipa e das entregas dos projetos.

Para você começar a olhar metodologias ágeis de uma forma mais séria, pesquisei e filtrei alguns benefícios que tirarão você da sua zona de conforto,

- A primeira é a garantia de entregas, quando você ouvir por metodologias ágeis, essa praticamente será o primeiro benefício que veras, Na verdade uma abordagem ágil permite organizar a equipa de projeto de forma a otimizar a sua flexibilidade, criatividade e produtividade. As tarefas são estimadas e divididas para serem incluídas num *sprint* (período de tempo no qual um conjunto de tarefas deve ser executado). Cada *sprint* tem uma duração fixa, o que faz com que a equipa fique focada num objetivo, tornando-a mais produtiva.

- A outra que julgo importante demais é o envolvimento do cliente, metodologias ágeis favorecem o envolvimento entre o cliente e a equipa de implementação do projeto, promovendo a colaboração, melhor compreensão da visão do cliente e a possibilidade de sugerir oportunidades de melhoria. Este envolvimento aumenta também a transparência e a confiança na capacidade de a equipa corresponder às expectativas do cliente. Acho esse benefício muito bom, porque muitas das vezes as equipes atrasam na entrega ou erram por causa de não falarem com o cliente.

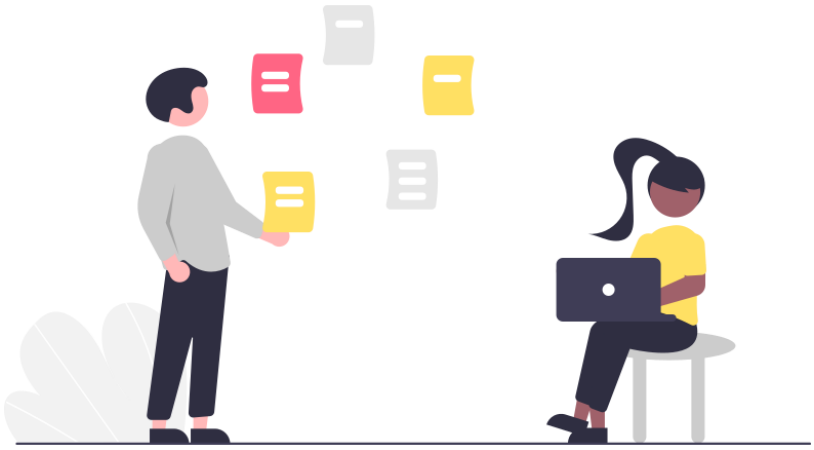
- O foco é o usuário, estas metodologias normalmente utilizam *user stories*, ou seja, descrições curtas e simples das funcionalidades, relatadas na perspectiva do utilizador que as vai executar. Deste modo, a equipa técnica consegue ter uma melhor compreensão necessidades do utilizador, facilitando a comunicação entre todos os interlocutores do projeto.

As metodologias ágeis são uma ferramenta poderosa para desenvolvimento de soluções tecnológicas. Permitem que a gestão do projeto seja mais controlada, nomeadamente nos seus custos ou alterações ao âmbito do projeto, além de proporcionarem ganhos importantes para os clientes e os seus negócios.

CAPÍTULO 15

MINHAS EXPERIÊNCIAS

“Seja bom na sua carreira**”.**



Estou muito feliz por voce chegar até esse tema, isso mostra que voce praticamente leu tudo que eu queria passar aqui, agora vou contar um pouco sobre minha carreira no mundo da programação, como iniciei, meus freelas, meu trabalho e tudo que passei até chegar aqui nesse ponto de escrever um e-book.

Acho que já contei essa parte da minha história, de como conheci e me tornei um programador, mais na verdade isso passa como um flashback, mais tudo isso aconteceu quando recebi o meu primeiro computador e fui ao curso básico de informática na ótica do utilizador, ali descobri em mim o meu futuro.

De la pra cá, estudei um monte de assuntos, iniciei meus estudos em programação antes mesmo de entrar no ensino médio, estudei algoritmos, linguagem C, quando entrei no ensino médio em 2017 encontrei a disciplina de TLP(Técnicas de Linguagens de Programação) foi a minha disciplina favorita posso confessar que deixei meu legado nessa disciplina amei programação mais a cada dia, aprendi C#, e com ela vendi o meu primeiro sistema de gestão para uma creche, foi algo muito motivador, fui aprendendo mais e mais.

Depois conheci a Web, eu gostava muito de desenvolver para desktop que me intitulava o homem que só trabalharia com Csharp para vida toda. Mais depois de um tempo conheci a web, já sabia antes da web, mais só a parte front-end, me aperfeiçoei mais, e conheci o meu amigo PHP.

O tempo passou, e estava na hora de viver de freelas, construir softwares para pessoas que quisessem, mas estava difícil para mim encontrar um freela, talvez pelo networking que é algo que quero aqui dizer se voce quer se tornar um dev moderno com oportunidades a suas atraz te perseguindo.

A minha estrada de trabalho começou nessa quarentena que as vezes me dá vontade de dizer que foi bom esse lance do covid-19 aparecer, enquanto estava de quarentena, eu precisava definir algumas coisas que queria realizar nesse ano, então um amigo me convidou para uma vaga de emprego como desenvolvedor fullstack, que na verdade desistimos por falta de seriedade no tal empreendedor, então eu notei que estava na hora de trabalhar e usar meus conhecimentos que aprendi depois de tanto tempo, e deixar de ficar em casa no meu local de conforto, então decidi filtrar vagas de emprego na minha conta do Facebook para programador ou

design, já que mexo também com design, de la pra cá, teve muitos contratempos, tive alguns freelas, e graças a Deus hoje trabalho em uma empresa como desenvolvedor web fullstack e mobile.

Se voce é iniciante nessa área, eu quero deixar um conselho que julgo bastante importante, é que voce deve definir os passos que deseja chegar daqui a um tempo, não é à toa que existe aquela pergunta que muitos chamam de motivação e de coachs que é “Onde voce se vê daqui a cinco anos”. Pode parecer clichê, mas não é.

Em cada ano eu defino que stacks ou tecnologias devo aprender, e acho que deveria fazer o mesmo. Porque nessa área da programação, existem muitas tecnologias e muita coisa para se aprender, e se você não ter um objetivo definido, voce vai se perder no meio de tanta coisa para aprender. Isso é muito conhecida como tecnologias pop stars, a maioria dos conteúdos apresentados aqui serão abordamos em uma playlist no meu canal do Youtube então, esteja à vontade em saber mais.

NETWORKING:

Talvez tinha que escrever um livro só com esse tema, porque é uma coisa muito importante para qualquer tipo de profissional, não so nos programadores, mais sim todos.

Networking é uma palavra inglesa que é traduzida como sendo a capacidade de estabelecer uma de contatos com pessoas, refere-se à troca de informações e conhecimentos com uma rede de contatos. O objetivo do *networking* é ampliar as oportunidades de sucesso profissional**.**

O Networking é muito importante que filtrei aqui algumas formas para voce fazer seus networkings

Mantenha as suas redes sociais atualizadas com suas informações profissionais.

Notou a frase final? Informações profissionais, repito profissionais, deixa um pouco aquela bobazeira das redes sócias de memes e outras coisas que não agregam nada, quer ter um bom networking e se destacar como um bom profissional, então voce precisa fazer um marketing pessoal e tudo começa nas redes sócias. Então acho que fui claro.

As outras dicas são bem obvias quando se fala de networking, participe de eventos e fóruns relacionados a sua área de atuação, retome o contato com as pessoas que conheceu em eventos. Demonstre interesse pelo

trabalho dos outros, faça boas perguntas, divulgue o seu trabalho nas redes sociais.

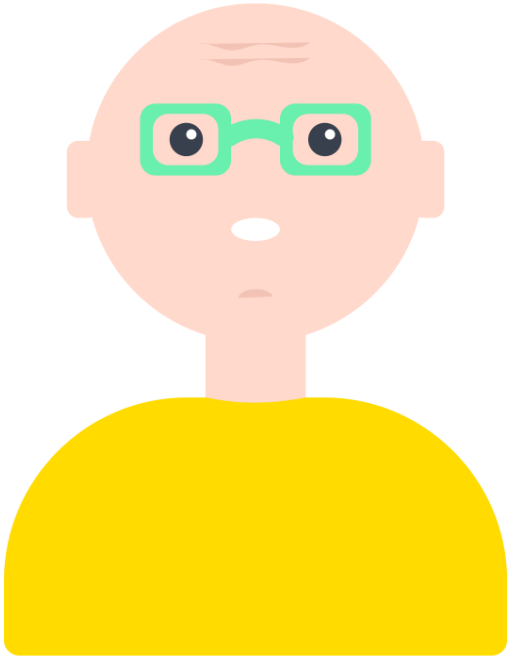
Essas dicas voce deve levar para sua vida toda se quer se tornar um desenvolvedor com uma carreira de sucesso, eu quando falo sobre networking gosto sempre de frisar e vender o peixe da comunidade que eu e um amigo meu que criamos, a CDA (communitydevangola), é uma comunidade de programadores, lá tem de tudo, web, desk, mobile, design, tem de tudo quanto é profissional, então voce precisa conhecer pessoal desse tipo para teres mais feedbacks.

Se deseja saber mais sobre meu início e aprender mais comigo, pode me contatar por e-mail, e será um prazer falar consigo e lançar um vídeo no meu canal para falar mais do meu início na programação.

CAPÍTULO 16

DICAS DE OURO

"Programar é a ciência do novo futuro**".**



Bem, terminamos de falar das stacks e feactures que eu queria vos dar a conhecer que um dev moderno deve aprender, mais eu quero deixar aqui um número simbólico de dicas.

Essa vai para os iniciantes em uma determinada linguagem de programação, “Nunca apreenda nenhum framework, sem entender a linguagem”.

Primeiro é que irás se frustrar, se você começa pelo Bootstrap sem passar no CSS, você está caminhando mal, se começares por Laravel sem passar por aquela dor de cabeça que o MVC e o PHP, você está começando mal, se começar por aprender frameworks para consumir APIs e não sabes consumir com o puro, está a começar errado, aprenda o puro, depois com o tempo, você aprende os frameworks, e aí você vai notar que eles estão te ajudando.

Outra dica, vai para todos devs em geral, seja iniciante, intermediário e o avançado, nunca aprenda uma stack porque outro dev aprendeu, ou você viu todos a aprenderem e também queres aprender. Saiba porque quer aprender uma determinada tecnologia, senão vai se frustrar, saiba ver o que ela vai agregar e acompanhe o que o mercado de trabalho está procurando, esse talvez seja o seu único motivo em escolher uma stack para aprender.

Outra vai para programadores autodidata e não autodidata, pesquise 7x7, procure, saiba porque existe, mas sempre seja pesquisador de stacks novas, se atualize, ame o que você faz. E eu sei que amas, senão não estarias a ler este livro.

Outra dica vai para os dev moderno com muito conhecimento, seja um evangelista, não falo sobre religião, o evangelista na programação é o cara que ensina aos outros, partilha o que sabe, participa em eventos do tipo, seja mentor de alguém, escreva um livro, escreva um blog, mas espalhe o que você sabe, assim além de conhecimento que vás adquirir, respeito na área, e muitas oportunidades também serão as consequências dessa prática.

CONCLUSÃO

Escrever um livro nunca foi fácil, você tem de saber o que você está a escrever para conseguir passar seus conhecimentos aos outros, além de você ensinar, você aprende muito escrevendo, é uma coisa que notei enquanto escrevia essas pequenas páginas

Nesse momento, podemos concluir se assim ainda posso dizer, que o dev moderno é um autodidata, alguém focado e em constante pesquisa sobre desenvolvimento, nós falamos aqui sobre muita coisa, como Padrões de Projeto, Orientação a Objeto, Padrões de Arquitetura MVC, Web Services APIS, Git e Github, Metodologias Ágeis, Segurança, Web semântica, plataformas web e desktop, falamos quase de tudo, o que eu escrevi no meu rascunho sobre como eu penso e deve ser um dev moderno foi passado para vocês, ainda tem muita coisa pra falar sobre como API REST, DevOPS, TDD, DDD, as formas de testes, muita coisa que vou tentar levar noutras edições do livro.

Obrigado e espero que tenhas gostado do livro.

Toda crítica, elogios, você me envia nos meus contatos que se encontram no princípio do livro.

Obrigado caro leitor...

REFERÊNCIAS

Imagens dos títulos:

www.undraw.co – Site onde se baixou imagens para representação dos temas.

Significados de Palavras Técnicas:

Open-source: Código livre, palavra técnica sobre códigos livres.

Stacks : Tecnologias ou ferramentas usadas para o desenvolvimento de softwares

Dev: Abreviação para Desenvolvedor de Software

UXDesign: Uxperience Design (Experiência do Design)

UIDesign: User Interface (Interface do Usuário)

Git: Programa de versionamento de código

Github: Plataforma Web, conhecida como repositório ou rede social de para programadores