pricefyLabs

# Arquitetura de Software 101

## O mínimo que você precisa saber

**Leandro Silva**, JAN/22


pricefyLabs

https://medium.com/pricefy-labs

## Estamos contratando para os times de engenharia e produtos.

# vamos conversar?

**ProdOps - Engenharia e Produto com Leandro Silva**

https://www.youtube.com/watch?v=2IxX2f0ZckQ

https://www.youtube.com/watch?v=jOeuK2U8vI8

**ElvenWorks - Conhecendo a tecnologia por trás de uma solução muito inteligente de Precificação**

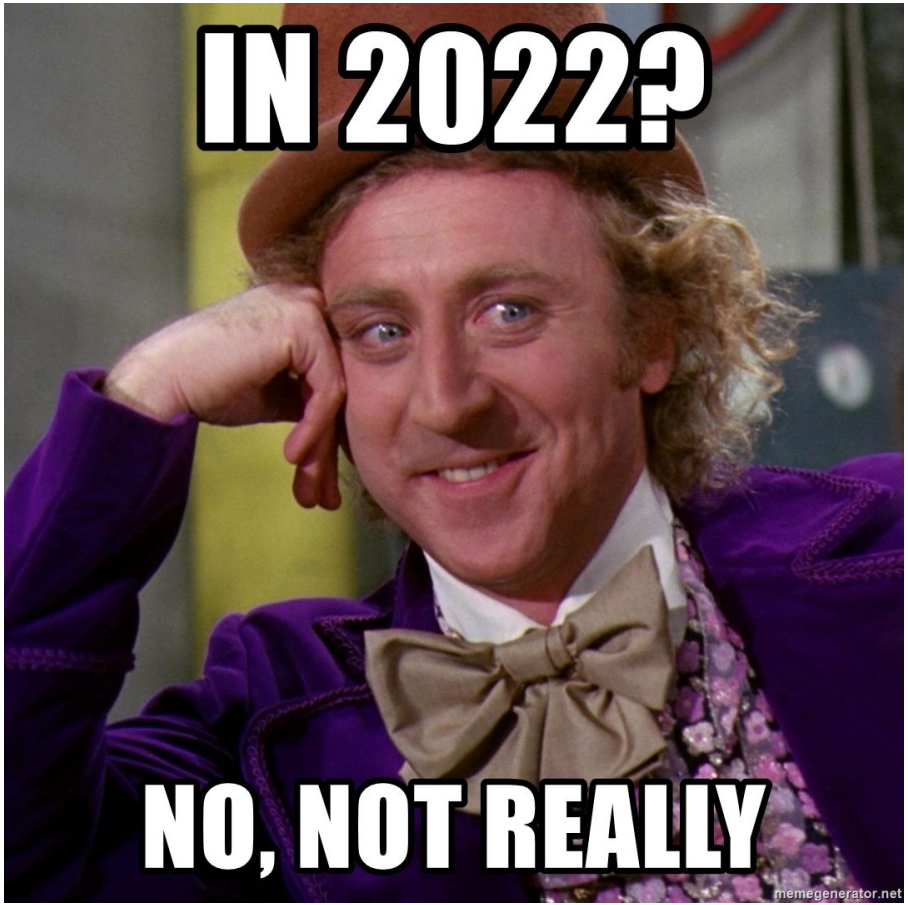
https://www.youtube.com/watch?v=DCT0IoRcrUo

# O que é arquitetura?

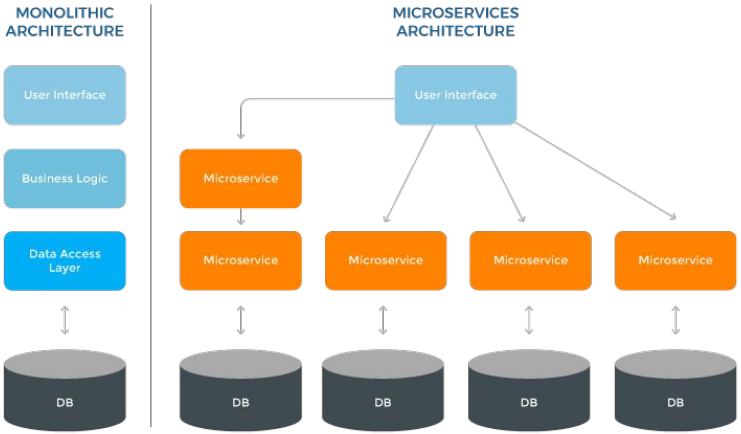
“Arquitetura de software é sobre fazer as escolhas estruturais fundamentais, que são caras de mudar depois de implementadas.”

- Software architecture, Wikipédia

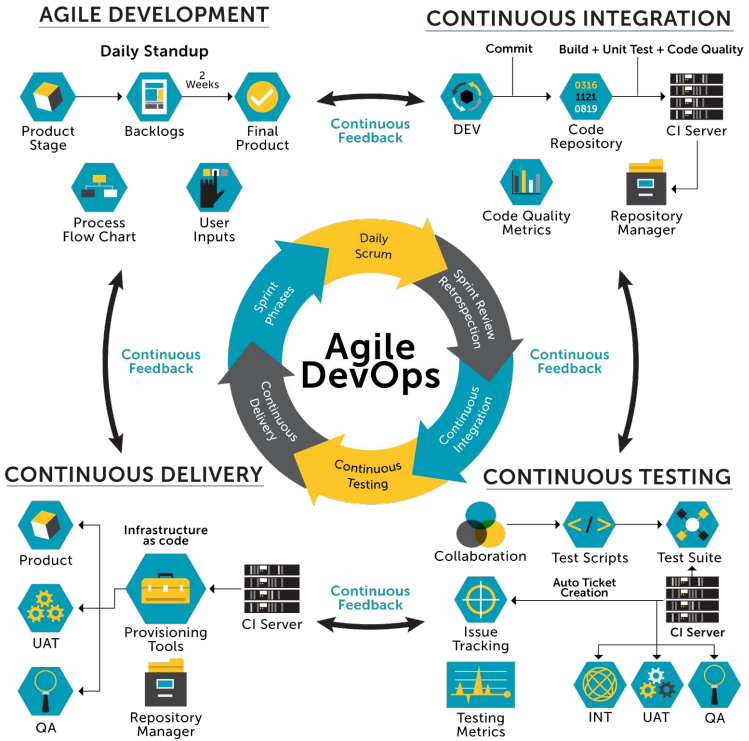
https://en.wikipedia.org/wiki/Software_architecture


IN 2022?
NO, NOT REALLY
memegenerator.net

2005


MONOLITHIC ARCHITECTURE
User Interface
Business Logic
Data Access Layer
DB
MICROSERVICES ARCHITECTURE
User Interface
Microservice
Microservice
Microservice
Microservice
Microservice
DB
DB
DB
DB
DB

2009


AGILE DEVELOPMENT
Daily Standup
Product Stage
Backlogs
2 Weeks
Final Product
Process Flow Chart
User Inputs
CONTINUOUS INTEGRATION
Commit
Build + Unit Test + Code Quality
DEV
0316 1121 0819
Code Repository
CI Server
Code Quality Metrics
Repository Manager
Continuous Feedback
Continuous Feedback
Continuous Feedback
Agile DevOps
Daily Scrum
Sprint Review Retrospection
Continuous Integration
Continuous Testing
Continuous Delivery
Sprint phrases
CONTINUOUS DELIVERY
Product
UAT
QA
Infrastructure as code
Provisioning Tools
CI Server
Repository Manager
CONTINUOUS TESTING
Collaboration
Test Scripts
Test Suite
Auto Ticket Creation
Issue Tracking
CI Server
Testing Metrics
INT
UAT
QA

Com os adventos **microsserviços** e **devops**, mudanças arquiteturais já não são tão rígidas e custosas quanto costumavam ser; e ainda podem ser feitas incrementalmente.

## Lição #1

O ecossistema evolui e muda constantemente, assim também o **contexto** no qual as **decisões** são tomadas.

Okay. Vejamos mais algumas definições então...

“Além dos algoritmos e estruturas de dados de computação, projetar e especificar a estrutura geral do sistema surge como um novo tipo de problema. As questões estruturais incluem organização bruta e estrutura de controle global; protocolos para comunicação, sincronização e acesso a dados; atribuição de funcionalidade aos elementos de design; distribuição física; composição de elementos de design; escalabilidade e performance; e seleção entre alternativas de projeto.”

**- An Introduction to Software Architecture, David Garlan & Mary Shaw**

https://userweb.cs.txstate.edu/~rp31/papers/intro_softarch.pdf

“A concepção de mais alto nível de um sistema em seu ambiente. A arquitetura de um sistema de software (em um determinado momento) é sua organização ou estrutura de componentes importantes interagindo por meio de interfaces; esses componentes sendo composto de componentes e interfaces sucessivamente menores.”

**- IEEE98 + RUP**

“Na maioria dos projetos de software bem-sucedidos, os desenvolvedores especialistas que trabalham nesse projeto têm um entendimento compartilhado do projeto do sistema. Esse entendimento compartilhado é chamado de "arquitetura". Esse entendimento inclui como o sistema é dividido em componentes e como os componentes interagem por meio de interfaces. Esses componentes geralmente são compostos por componentes menores, mas a arquitetura inclui apenas os componentes e interfaces que são compreendidos por todos os desenvolvedores.”

**- Extreme Programming mailing list, Ralph Johnson**

https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf


COULD YOU BE
ANY MORE VAGUE
makeameme.org

**Sendo assim, a definição que vamos seguir daqui em diante é...**

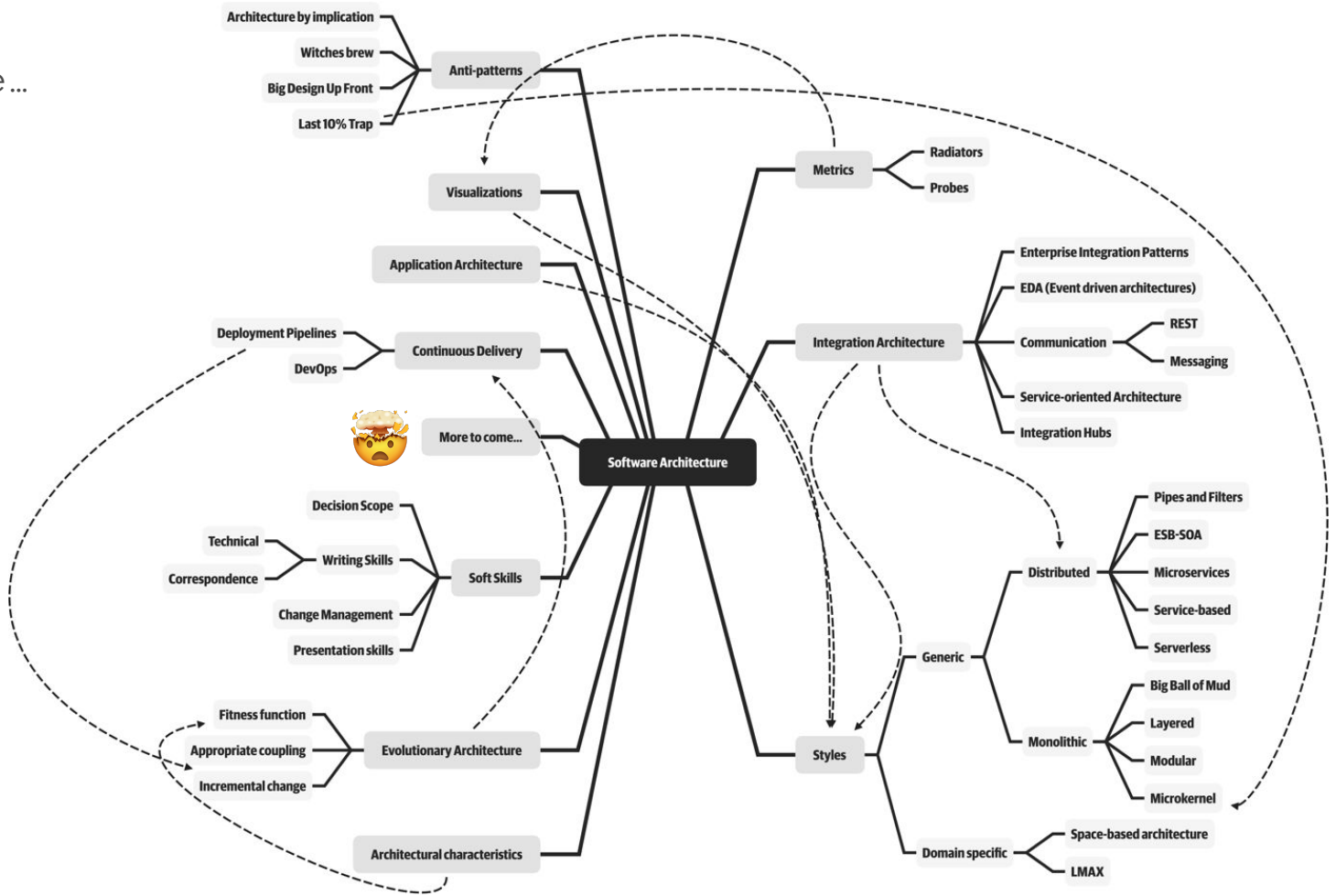
“Arquitetura é sobre as coisas importantes. Seja lá o que for.”

**- Ralph Johnson**

# Ser arquitet@ de software

# O que faz um arquitet@ de software?

# Lida com **"coisas importantes"**, uéâ€¦




Fundamentals of Software Architecture, Mark Richards & Neal Ford

WHEN YOU WANNA BE A SOFTWARE ARCHITECT


BUT CAN'T DEAL WITH ALL THE THINGS ALONE

Nem todas as coisas são importantes; e mesmo as que são, não são o tempo todo.

**Lição #2**

As **“coisas importantes”** são as coisas que são importantes para o **time**.

Time? Oi?

TIME equipe

"Arquitetura é uma **construção social** (bem, software também é, mas arquitetura é ainda mais), porque não depende apenas do software, mas de qual parte do software é considerada importante pelo consenso do grupo."

"Se algo faz parte da arquitetura é inteiramente baseado em se os desenvolvedores acham que é importante."

**- Ralph Johnson**

https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

Thanks. Eu não preciso de arquitet@ no time.

design
Editor: Martin Fowler ■ ThoughtWorks ■ fowler@acm.org

# Who Needs an Architect?

Martin Fowler

"Esse tipo de arquiteto deve estar muito atento ao que está acontecendo no projeto, procurando por questões importantes e abordando-as antes que se tornem um problema sério. Quando vejo um arquiteto assim, a parte mais perceptível do trabalho é a **intensa colaboração**."

Oryzus
>
Reloadus

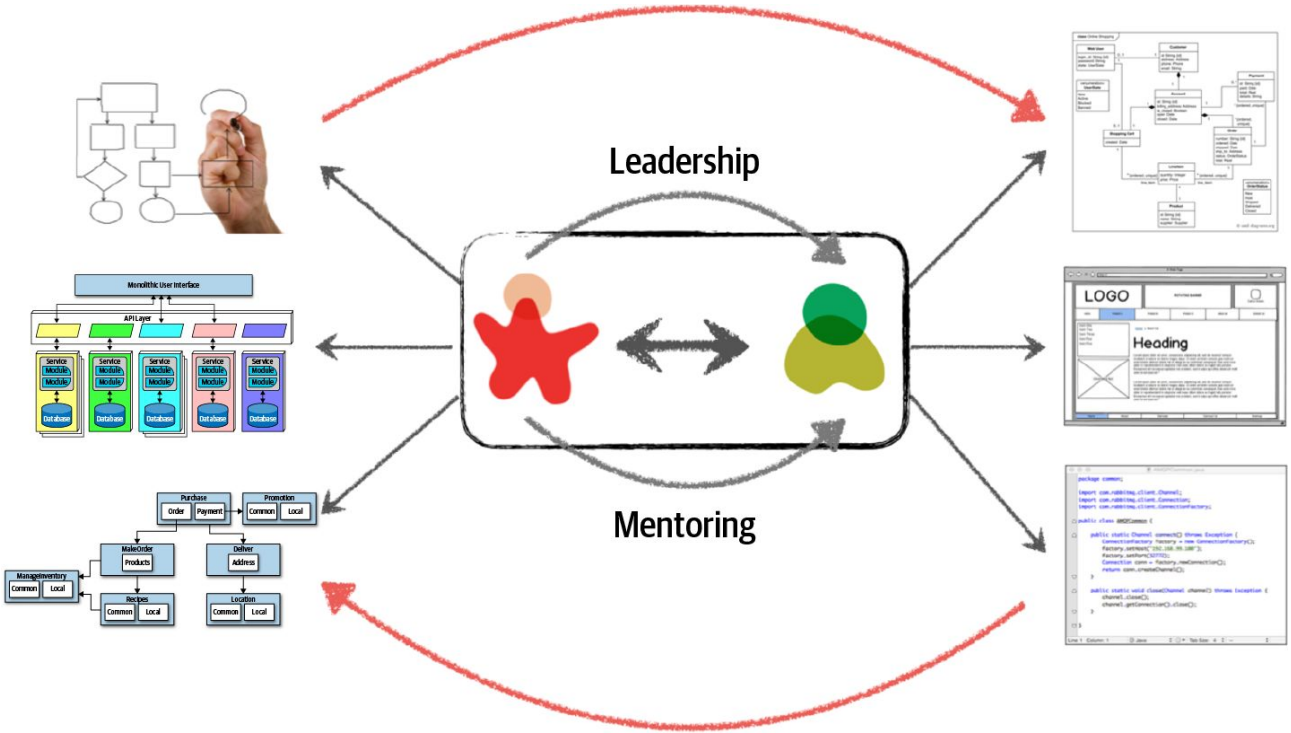
"De muitas maneiras, a atividade mais importante do **Architectus Oryzus** é orientar a equipe de desenvolvimento, elevar seu nível para que possam lidar com questões mais complexas. Melhorar a capacidade da equipe de desenvolvimento dá a um arquiteto uma alavancagem muito maior do que ser o único tomador de decisões e, portanto, correr o risco de ser um gargalo arquitetônico."

"Mike Two veio com o melhor nome que ouvi até agora: **guia**, como em montanhismo. Um guia é um membro da equipe mais experiente e habilidoso que ensina outros membros da equipe a se defenderem melhor, mas está sempre lá para as coisas realmente complicadas."

https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

Arquitetura

Desenvolvimento




Fundamentals of Software Architecture, Mark Richards & Neal Ford

Melhor do que ter um único cérebro centralizando toda a visão do sistema e tomando todas as decisões importantes sozinho **(Reloadus)**, de modo que o time não saiba nada além do próximo passo imediato, é ter um guia **(Oryzus)**, que trabalha colaborativamente com o time para resolver as questões mais importantes, cada qual a seu tempo.

## Lição #3

Guiar + Cooperar > Ditar + Controlar.

Vão esperar que você, arquitet@...

1. **Tome decisões de arquitetura e princípios de design**, junto com o time, apoiando-os conforme seu nível atual de maturidade;
2. **Continuamente analise e revise a arquitetura** em função do estado atual da tecnologia e do problema de negócio que se propõe a resolver;
3. **Se mantenha atualizado** com as últimas tendências de tecnologia e engenharia de software;
4. **Garanta a conformidade das decisões tomadas**, tanto em nível de arquitetura quanto de princípios de design;

5. **Diversifique sua exposição e experiência** com novas tecnologias, plataformas, frameworks, para ter insumo na hora de propor soluções e tomar decisões;

6. **Tenha conhecimento do domínio de negócio que está lidando**, de modo que possa elaborar soluções coerentes com tal;

7. **Cultive skills interpessoais**, porque você vai ter que trabalhar colaborando tanto com pessoal de negócio quando de engenharia, facilitando reuniões e liderando soluções;

8. **Compreenda e lide bem com questões políticas**, sendo capaz de navegar toda a organização, de cima a baixo.

Mas mais importante (IMHO) do que qualquer outra coisa...


IT'S A LITTLE SOMETHING
CALLED
TRADE OFF
memegenerator.net

# “Tudo em arquitetura de software é um **trade-off**.”

Neal & Richards costumam dizer que “se um arquiteto acha que descobriu alguma coisa que não é um trade-off, muito provavelmente, ele apenas não descobriu o trade-off **ainda**”.

Por isso o **“porque”** é mais importante do que o **“como”** - você pode olhar para um sistema e ver como foi feito, mas não que parâmetros embasaram cada decisão que o levou a ser como é.

The Laws of Software Architecture, Mark Richards & Neal Ford

“Programadores sabem os benefícios de todas as coisas e os trade-offs de nenhuma delas. Arquitetos precisam saber ambos.” - Rich Hickey

# Lição #4 Trade-offs. Conheça-os.

# Características de arquitetura

“Uma **solução de software** é resultado da tradução de requisitos de domínio para características de arquitetura e princípios de design.”

“A qualidade de um sistema é o grau em que o sistema satisfaz as necessidades declaradas e implícitas de seus vários stakeholders e, portanto, fornece valor.”

“Essas necessidades dos stakeholders (funcionalidade, desempenho, segurança, manutenibilidade etc.) são justamente o que está representado no modelo de qualidade, que categoriza a qualidade do produto em características e subcaracterísticas.”

**- ISO/IEC 25010**

https://iso25000.com/index.php/en/iso-25000-standards/iso-25010


https://iso25000.com/index.php/en/iso-25000-standards/iso-25010


BOSS SAID WE MUST HAVE IT ALL
I TAUGHT HIM THE MEANING OF TRADE-OFF.
memegenerator.net

# Não dá para ter tudo.

Um arquiteto tem que trabalhar junto aos stakeholders do projeto para determinar o que **realmente** é importante para o negócio.

| Preocupação de negócio | Característica de arquitetura |
|---|---|
| Fusão e aquisição | Interoperabilidade, escalabilidade, adaptabilidade, extensibilidade |
| Time to market | Agilidade, testabilidade, deployabilidade |
| Satisfação do usuário | Performance, disponibilidade, tolerância a falhas, testabilidade, deployabilidade, agilidade, segurança |
| Competitividade | Agilidade, testabilidade, deployabilidade, escalabilidade, disponibilidade, tolerância a falhas |
| Tempo e budget | Simplicidade, viabilidade |

Os **requisitos de negócio** dizem respeito, por exemplo, a número de usuários, número de transações simultâneas, volume de dados, sazonalidade, etc.

Exemplos clássicos de sazonalidade, com spike gigantesco de tráfego: **envio da declaração de IR**, **venda de ingressos** e **Black Friday**.

Tome **cuidado** com as predições que surgem de tais requisitos. Todo mundo tem uma ideia que vai atrair milhões de usuários.

Esteja alerta às **características implícitas**. Por mais que o requisito segurança possa não surgir como um requisito formal, ninguém quer usar um sistema inseguro, onde há vazamento de dados.

Exemplo de domínio de negócio influenciando implicitamente as características de uma arquitetura: um **sistema de imagens "médicas"** vai precisar de armazenamento e largura de banda muito maior de um **sistema de agendamento de consultas "médicas"**.

# Arquitetura menos pior.

Trade-offs


Uma arquitetura ~~nunca~~ **muito provavelmente** não vai ter todas as características que vimos até agora. **Não é viável.**

Por exemplo: **performance** e **segurança** se impactam mutuamente.

Pilar da engenharia ágil: **design iterativo**. Projete a arquitetura boa o bastante e incremente.

“Uma das diferenças entre **arquitetura de construção** e **arquitetura de software** é que muitas decisões sobre uma construção são difíceis de mudar. É difícil voltar e mudar seu porão, embora seja possível.

Não há nenhuma razão teórica para que algo seja difícil de mudar em software. Se você escolher qualquer aspecto do software, poderá facilitar a alteração, mas não sabemos como facilitar a alteração de tudo. Tornar algo fácil de mudar torna o sistema geral um pouco mais complexo, e tornar tudo fácil de mudar torna todo o sistema muito complexo. **Complexidade é o que torna o software difícil de mudar.** Isso e duplicação.”

Ouçam o tio **Ralph Johnson**, amiguinhos!

# Lição #5

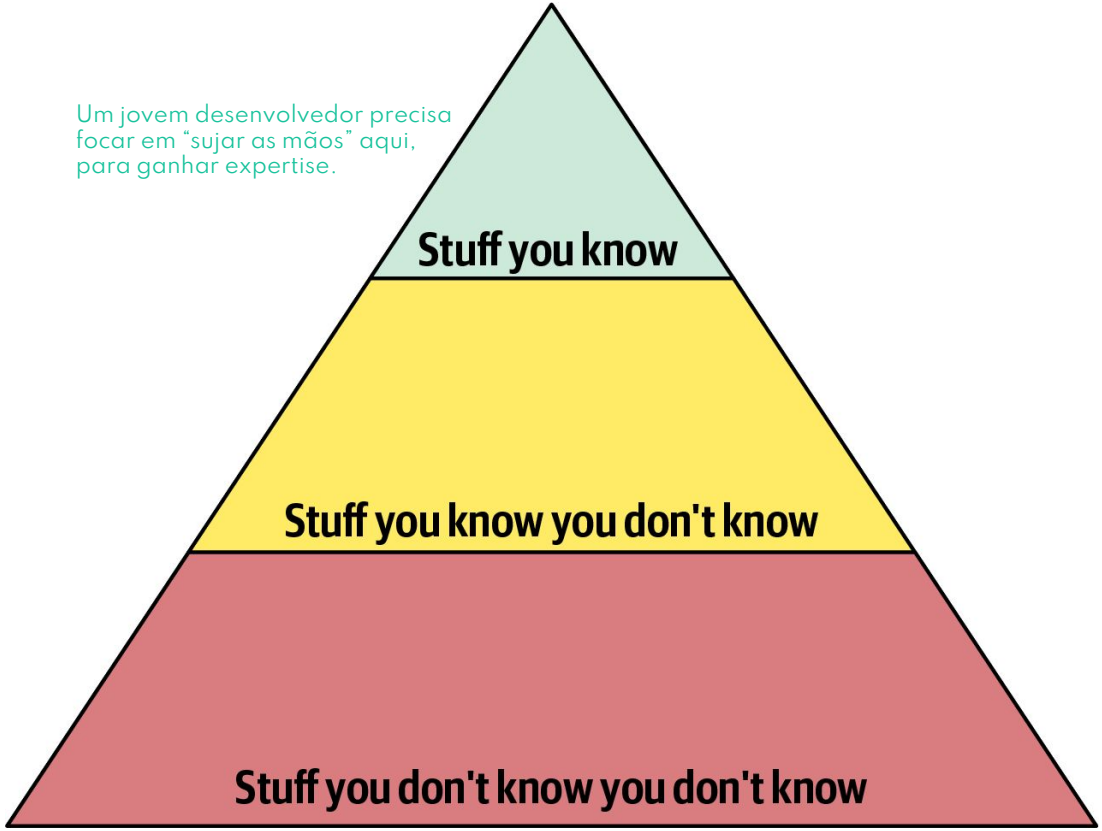
“O software não é limitado pela física, como os edifícios. É limitado pela imaginação, pelo design, pela organização. Em suma, é limitado pelos atributos das pessoas, não pelos atributos do mundo. **Encontramos o inimigo, e ele somos nós.**”

# Trade-offs de ser arquitet@ de software

“Assim como um meteorologista e um pintor veem uma nuvem de maneiras diferentes, assim um arquiteto em relação a outros profissionais da engenharia de software.”

Hmm, okay. Mas por que isso é importante?

Um jovem desenvolvedor precisa focar em "sujar as mãos" aqui, para ganhar expertise.

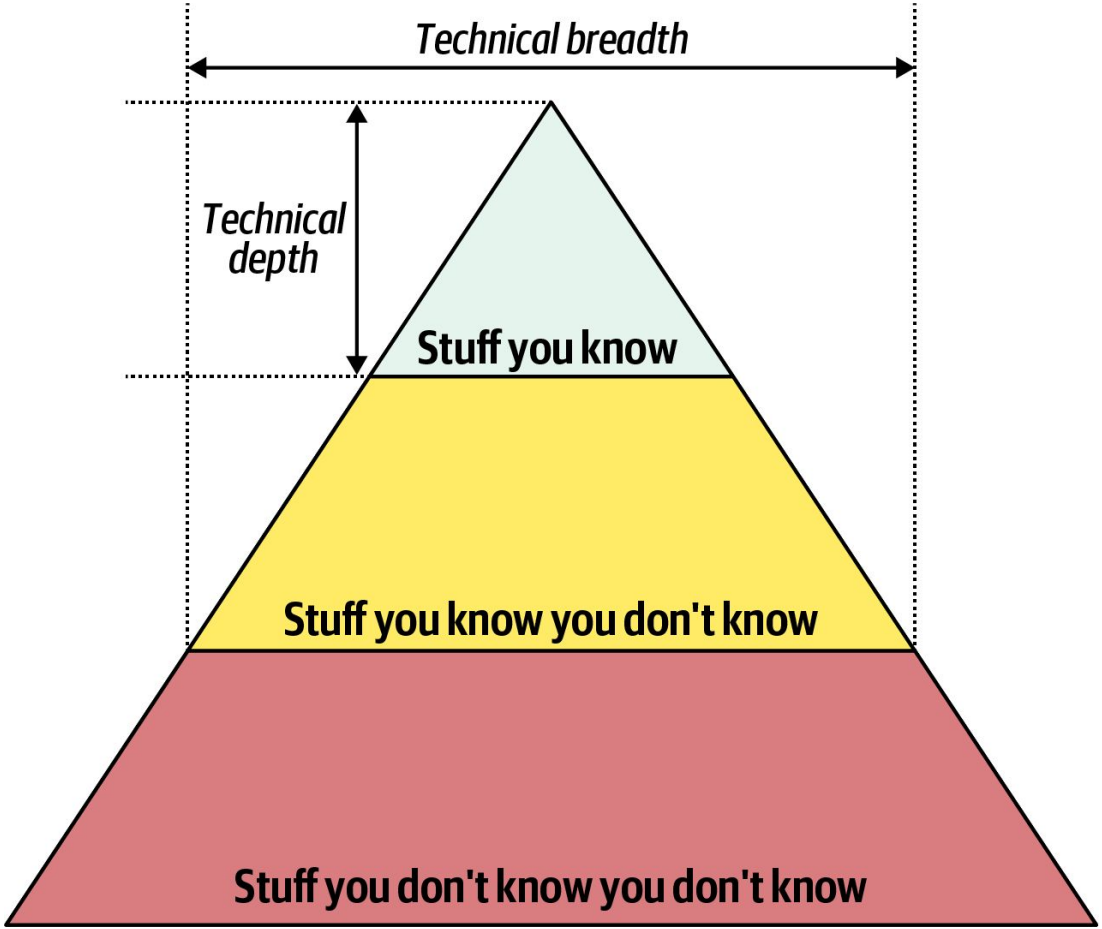



Linguagens, frameworks, bibliotecas, ferramental de debugging, coisas que você usa no seu dia a dia para fazer o seu trabalho de desenvolvimento de software.

O que você conhece superficialmente ou pelo menos já ouviu falar (ex.: Brainfucker, eu sei o que é, mas nunca escrevi sequer um hello world).

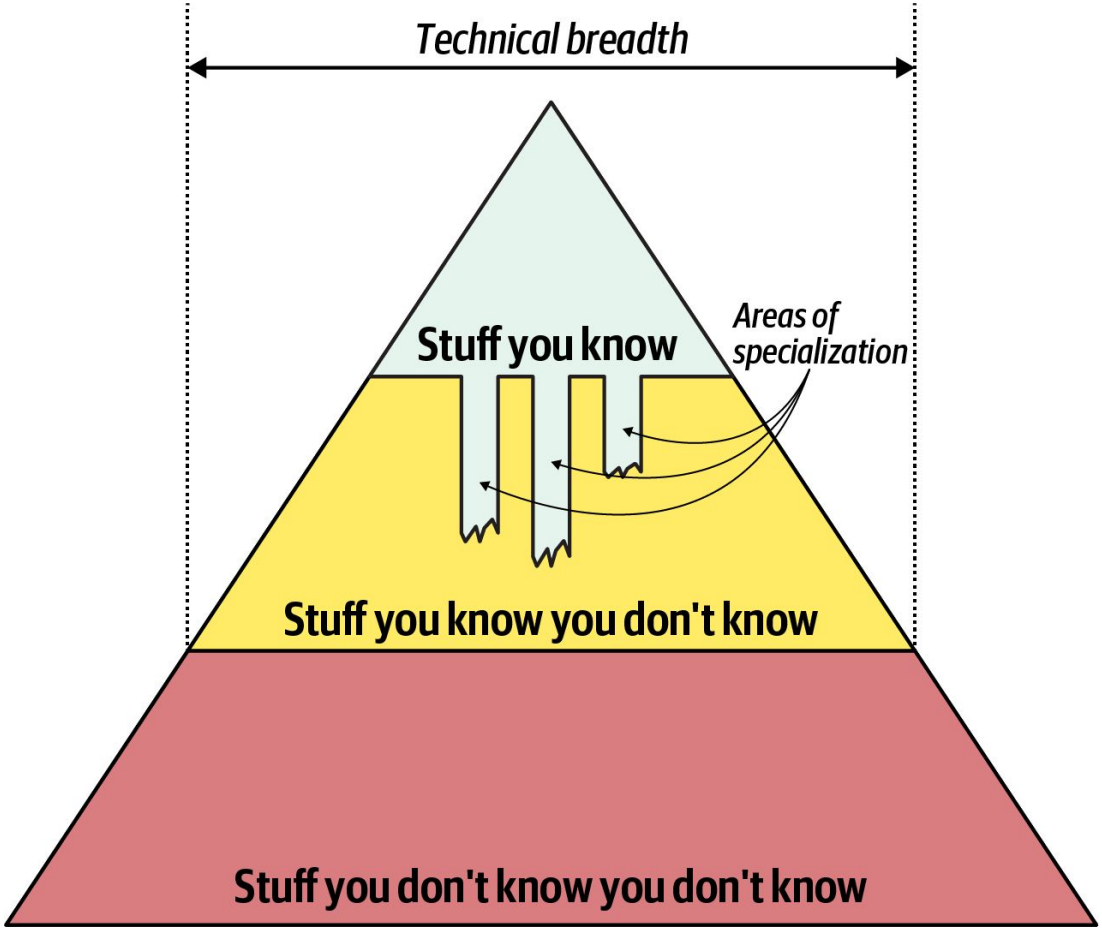
É uma gigantesca quantidade de coisas que poderiam solucionar perfeitamente o seu problema à mão, mas que você sequer faz ideia de que elas existem.

Fundamentals of Software Architecture, Mark Richards & Neal Ford




Para um arquiteto, technical breath é mais importante do que technical depth. Porque um arquiteto precisa tomar decisões que combinam **"o que é necessário"** com **"o que é possível"** (requisitos vs. restrições), então ter conhecimento tecnológico largo é mais importante do que ter conhecimento profundo.

Fundamentals of Software Architecture, Mark Richards & Neal Ford




**Não dá para ser expert em tudo.** Você tem que saber onde precisa manter profundidade **(áreas de especialização)** e onde pode sacrificar profundidade para ganhar largura.

Fundamentals of Software Architecture, Mark Richards & Neal Ford

1. Alargar o conhecimento
2. Se manter atualizado em relação às tendências mais atuais de tecnologia
3. Estar bem consciente do contexto atual

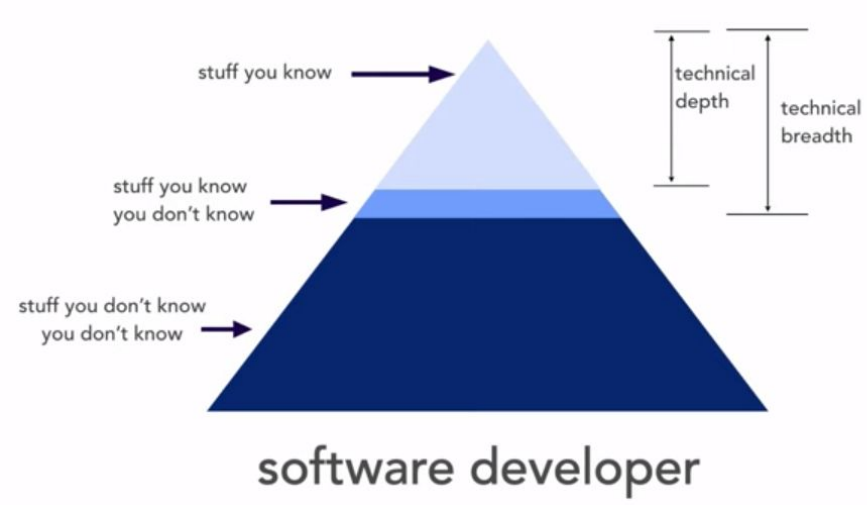

# Frozen Caveman Anti-Pattern

"Eu tenho uma tonelada de experiência usando SOA e ESBs em larga escala, por isso estou propondo essa solução que incorpora essas tecnologias"

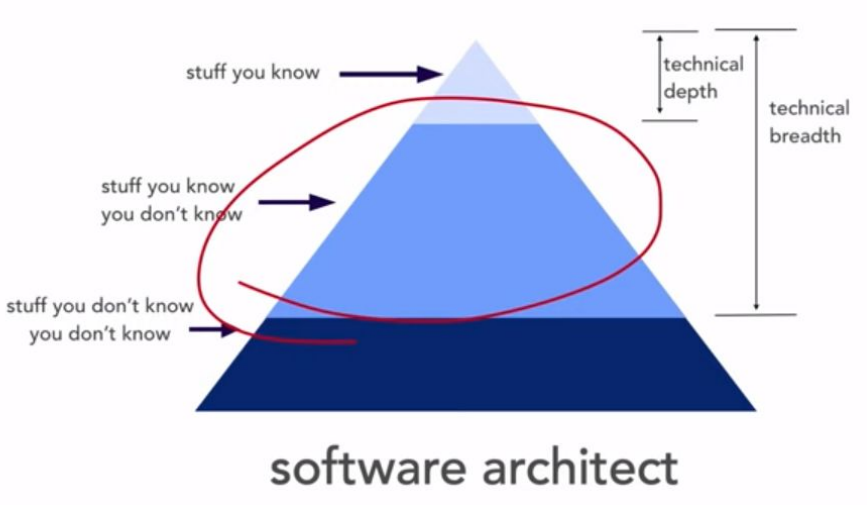
"Eu nunca ouvi falar em ReactJS, então, para essa solução, vamos ser conservadores e usar frameworks bem estabelecidos: JSP e Struts."

"Eu já me queimei muitas vezes no passados com processamento assíncrono, usando eventos e mensageria, então, para essa solução vamos usar somente REST."

https://www.youtube.com/watch?v=yK9b1C97ct0




software developer




software architect

https://www.youtube.com/watch?v=yK9b1C97ct0

# Praticando arquitetura

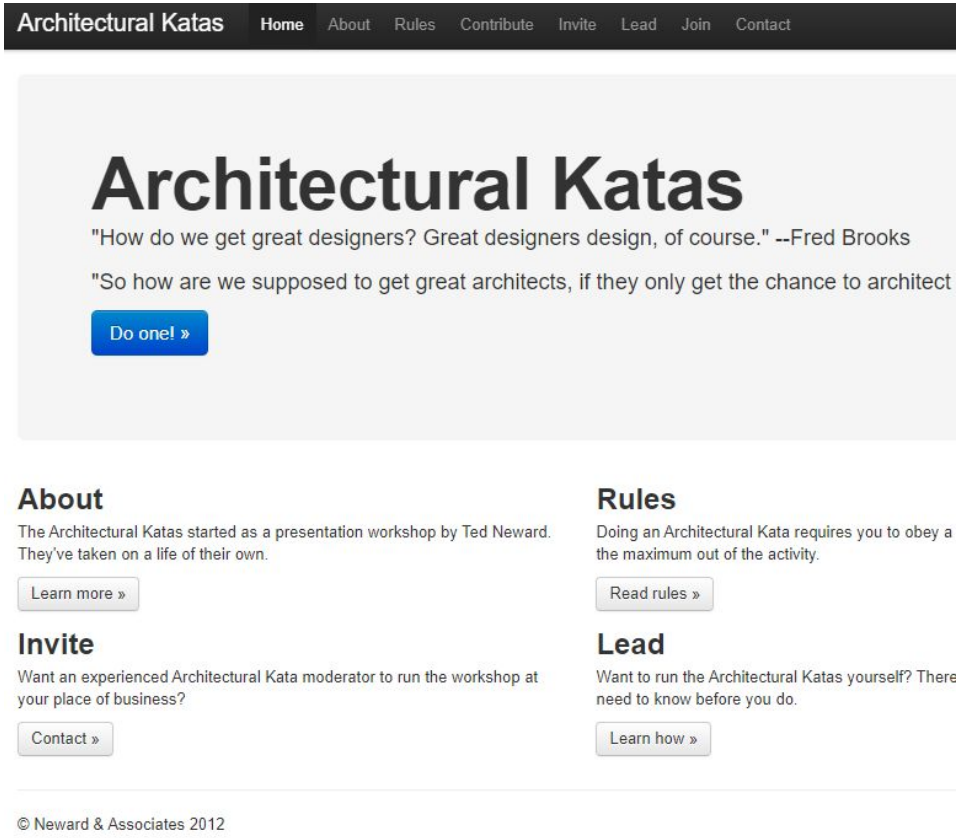
“Como conseguimos grandes designers de software? Grandes designers de software projetam, é claro.”

- Fred Brooks

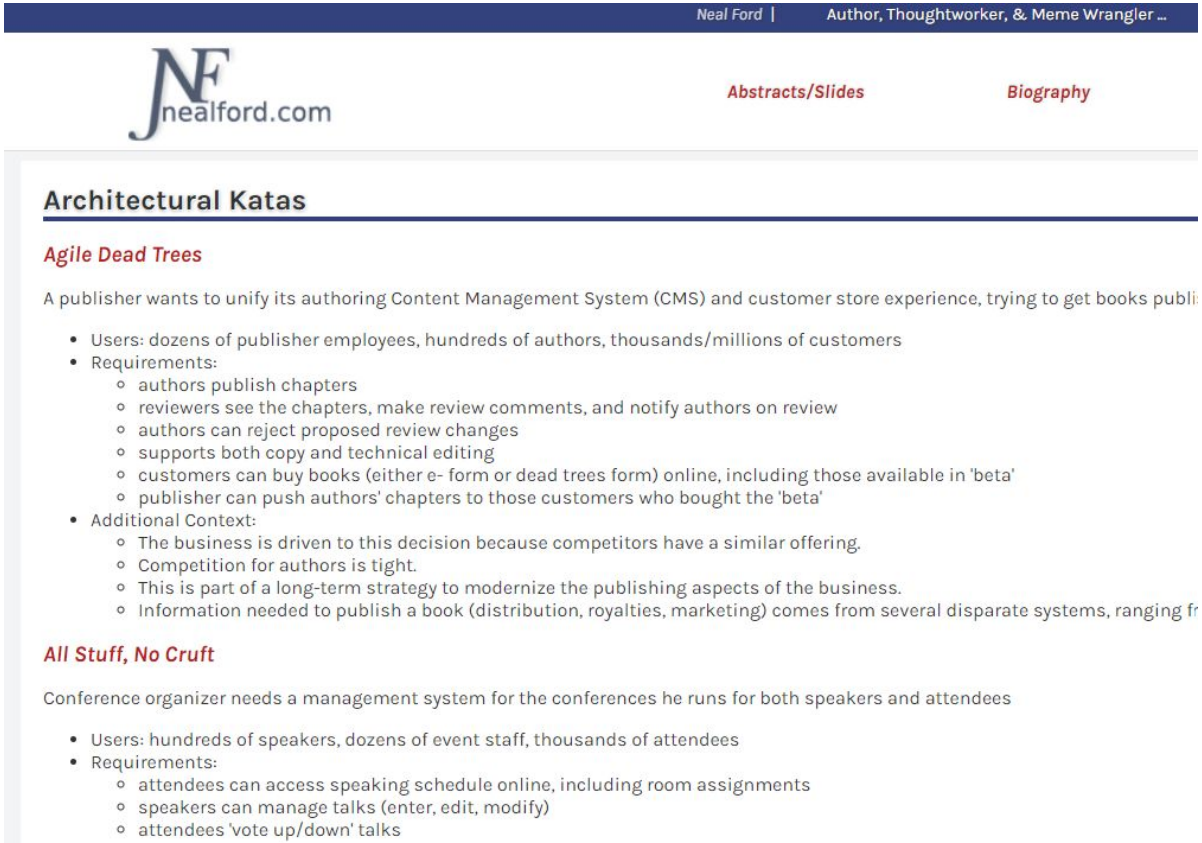
“Então, como podemos obter grandes arquitetos de software, se eles só têm a chance de arquitetar menos de meia dúzia de vezes em sua carreira?”

- Ted Neward

https://nealford.com/katas/




https://archkatas.herokuapp.com/



Neal Ford | Author, Thoughtworker, & Meme Wrangler ...

nealford.com

Abstracts/Slides Biography

## Architectural Katas

### *Agile Dead Trees*

A publisher wants to unify its authoring Content Management System (CMS) and customer store experience, trying to get books publi

- Users: dozens of publisher employees, hundreds of authors, thousands/millions of customers
- Requirements:
  - authors publish chapters
  - reviewers see the chapters, make review comments, and notify authors on review
  - authors can reject proposed review changes
  - supports both copy and technical editing
  - customers can buy books (either e- form or dead trees form) online, including those available in 'beta'
  - publisher can push authors' chapters to those customers who bought the 'beta'
- Additional Context:
  - The business is driven to this decision because competitors have a similar offering.
  - Competition for authors is tight.
  - This is part of a long-term strategy to modernize the publishing aspects of the business.
  - Information needed to publish a book (distribution, royalties, marketing) comes from several disparate systems, ranging fr

### *All Stuff, No Cruft*

Conference organizer needs a management system for the conferences he runs for both speakers and attendees

- Users: hundreds of speakers, dozens of event staff, thousands of attendees
- Requirements:
  - attendees can access speaking schedule online, including room assignments
  - speakers can manage talks (enter, edit, modify)
  - attendees 'vote up/down' talks

https://nealford.com/katas/

# Se aprofundando no assunto

# Recomendações

- Software architecture - Wikipedia
  https://en.wikipedia.org/wiki/Software_architecture
- An Introduction to Software Architecture - David Garlan & Mary Shaw
  https://userweb.cs.txstate.edu/~rp31/papers/intro_softarch.pdf
- Who Needs an Architect? - Martin Fowler
  https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf
- ISO/IEC 25010
  https://iso25000.com/index.php/en/iso-25000-standards/iso-25010
- Software Architecture Monday - Mark Richards
  https://www.youtube.com/playlist?list=PLdsOZAx8I5umhnn5LLTNJbFgwA3xbycar

# Recomendações

- The Mythical Man-Month - Fred Brooks
- Architecting for Scale - Lee Atchison
- Fundamentals of Software Architecture - Mark Richards & Neal Ford
- Software Architecture: The Hard Parts - Neal Ford, Mark Richards, Pramod Sadalage, Zhamak Dehghani
- Release It - Michael T. Nygard
- The Art of Scalability - Martin L. Abbott
- Designing Data-Intensive Applications - Martin Kleppmann
- Patterns of Enterprise Application Architecture - Martin Fowler

# Recomendações

- Enterprise Integration Patterns - Gregor Hohpe & Bobby Woolf
- Software Architecture Patterns - Mark Richards
- Handbook of Software Reliability Engineering - Michael R. Lyu
  http://www.cse.cuhk.edu.hk/~lyu/book/reliability/
- Architectural Katas - Ted Neward
  https://archkatas.herokuapp.com/
- Architectural Katas - Neal Ford
  https://nealford.com/katas/
- The System Design Primer - Donne Martin
  https://github.com/donnemartin/system-design-primer

# Obrigado!


medium.com/pricefy-labs

github.com/leandrosilva

leandrosilva.com.br