

# **O que gostaria de ter aprendido quando comecei uma carreira em TI?**

Edward Ribeiro

# \$ whoami

- Servidor Público
  - Serpro, STM, TSE, CL-DF, Senado (atualmente)
- Software Engineer - DataStax (2014/2015)
  - DSE Search
- Pesquisador independente (UnB)
  - Sistemas Distribuídos
- Contribuidor (eventual) de software livre
  - Cassandra, Solr, Kafka, ZooKeeper, VoltDB, etc
- ~~Professor Universitário~~





# Sua carreira deve ser planejada

- *Essa profissão é um eterno aprendizado (mesmo!);*
- A zona de conforto é igual a estagnação;
- **Essa atividade recompensa a prática;**



# O preço da estagnação

***15 anos de experiência  
ou 1 ano de experiência  
repetido 15 vezes?***



# Pratique, pratique, pratique

- Projetos de final de semana;
- Sites de programação competitiva ou exercícios (HackerRank, exercism);
- Projetos open source;
- *ABC (Always Be Coding)*
  - tradutores, bancos de dados, servidores web, aplicativos móveis, rede social, etc





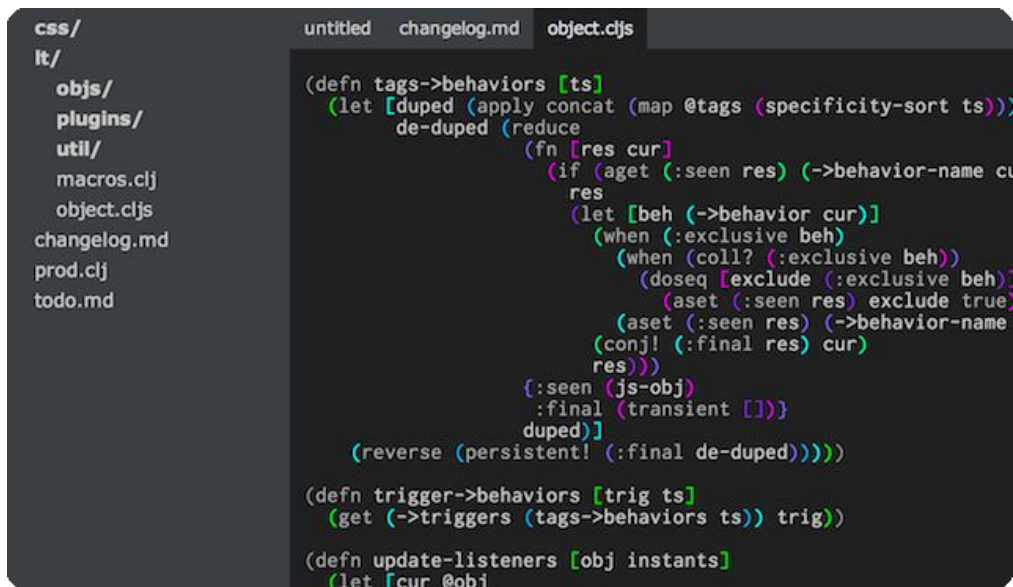
# Colabore com software livre

- Excelente aprendizado;
- Irá lhe tornar um programador melhor;
- É a chance de trabalhar num projeto real, usado por inúmeras pessoas;
- Tutoria de graça! Com os melhores!
- Constrói um currículo e abre oportunidades de emprego;
- Fazer parte de uma comunidade;



# Leia muito código

- Código é lido  **muito mais vezes**  que escrito!
- Leitura de código favorece:
  - Eficiência em dominar novas bases de códigos;
  - Domínio da linguagem e de construções idiomáticas;
  - Aprendizado de algoritmos e estruturas de dados, design de software;
- **Existe uma infinidade de software livre para ler**
- **Torne isso um hábito!**

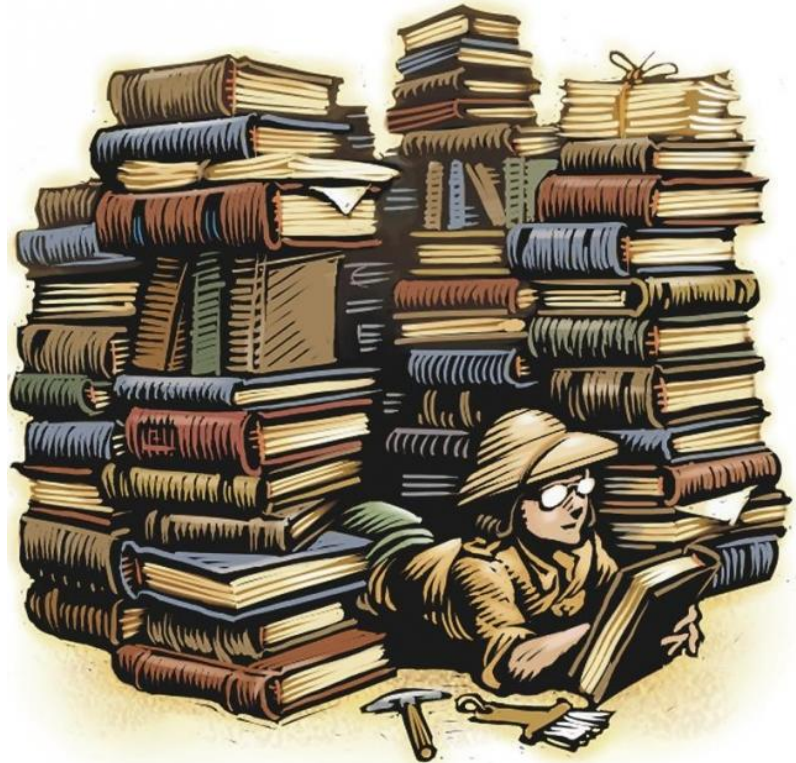


```
css/  
lt/  
  objs/  
  plugins/  
  util/  
  macros.clj  
  object.cljs  
  changelog.md  
  prod.clj  
  todo.md  
untitled  changelog.md  object.cljs  
  
(defn tags->behaviors [ts]  
  (let [duped (apply concat (map @tags (specificity-sort ts)))  
        de-duped (reduce  
          (fn [res cur]  
            (if (aget (:seen res) (->behavior-name cur)  
                res  
                (let [beh (->behavior cur)]  
                  (when (coll? (:exclusive beh))  
                    (doseq [exclude (:exclusive beh)]  
                      (aset (:seen res) exclude true)  
                      (aset (:seen res) (->behavior-name  
                            (conj! (:final res) cur)  
                            res))))  
                  { :seen (js-obj)  
                    :final (transient [])  
                    duped })  
                  (reverse (persistent! (:final de-duped))))))  
        ])  
    (reverse (persistent! (:final de-duped)))))  
  
(defn trigger->behaviors [trig ts]  
  (get (->triggers (tags->behaviors ts)) trig))  
  
(defn update-listeners [obj instants]  
  (let [cur @obj
```



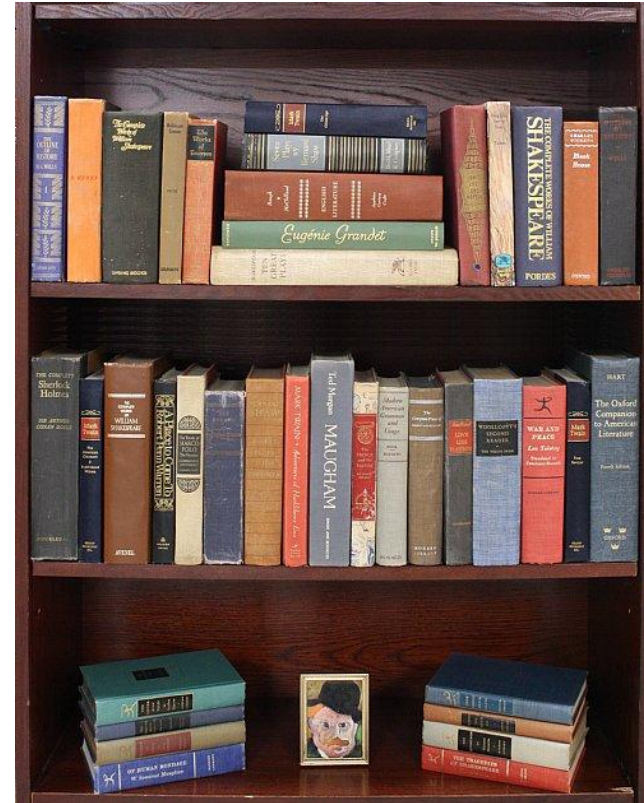
# Seja um leitor ávido

- Abra seus horizontes para assuntos fora da área de TI;
- Combustível para criatividade, e inspiração;
- Seja curioso



# Leia os clássicos

- Clean Code - Bob Martin
- Refactoring - Martin Fowler
- Working effectively with legacy code - Michael Feathers
- Domain Driven Design - Eric Evans
- Patterns of Enterprise Application Architecture - Martin Fowler
- Release It! - Michael Nygard
- A philosophy of Software Design - John Ousterhout
- Code Simplicity - Max Kanat Alexander
- etc



# **Livros!**

- **Apprenticeship Patterns: Guidance for the Aspiring Software Craftsman**
- **The Passionate Programmer**
- **Becoming a Better Programmer**
- **The Pragmatic Programmer**
- **Soft Skills**
- **Remote: No Office Required**

# Se preparando para entrevistas

- Cracking the Code Interview (Livro)
- <https://www.careercup.com>
- <https://www.hackerrank.com>
- <https://www.exercism.io>
- <https://leetcode.com>

# Domine as ferramentas de trabalho

- IDEs (shortcuts, plugins, ambiente);
- Controle de versão (Git e Github);
- Serviço de integração contínua;
- Linguagens de script (Bash, Zsh);
- Comandos Linux: ***grep, find, xargs, awk, sed, etc;***
- Conheça muito bem as APIs de sua linguagem principal;



# Não se apegue demais a uma única linguagem

- Domine mais de uma linguagem;
- Não existe linguagem perfeita nem onipresente;
- O aprendizado de novas linguagens lhe permitirá conhecer novos paradigmas, conceitos e melhores formas de resolver problemas.
  - Ex: Programação Funcional





# Não seja perfeccionista

- **Não existe software perfeito**
- O perfeccionista preza pela qualidade;
- Mas o perfeccionista dificilmente entrega algo (ou no prazo);
- Saiba quando parar (“bom o suficiente”) e lançar o sistema;



# Automatize tarefas

- Automatize tarefas cotidianas e repetitivas;
- Busque formas de tornar seu trabalho mais eficiente e evitar distrações com tarefas secundárias;
- Foco em produtividade;
- ***Seu workflow de trabalho deve ser automatizado também***



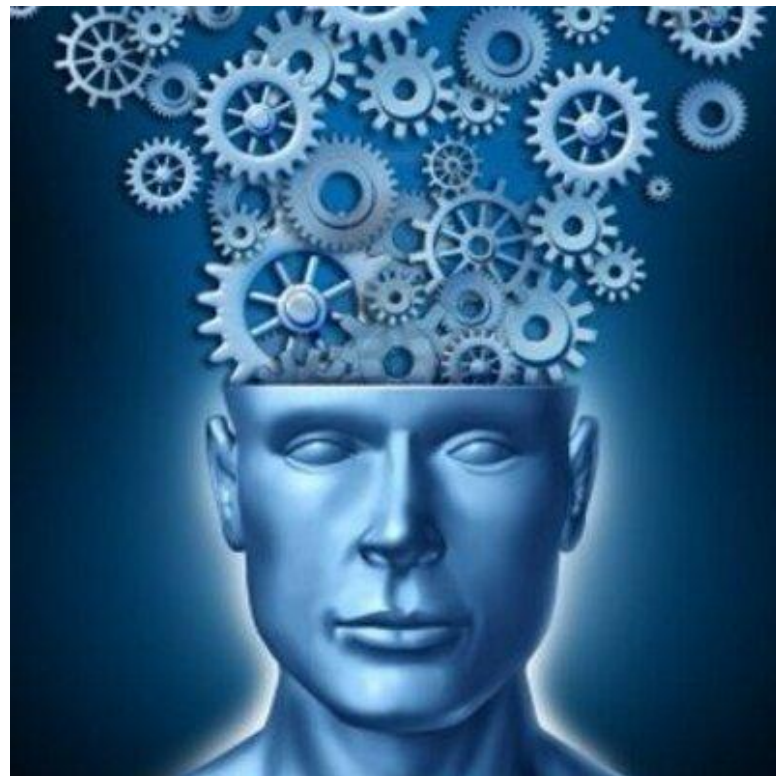
# Evite a estagnação

- Este campo avança muito rapidamente;
- Nos 10 últimos anos tivemos:
  - Cloud computing
  - Big data
  - NoSQL/NewSQL
  - (Real time) Analytics
  - Arquiteturas Orientadas a Microsserviços
  - Serverless
  - Blockchain
  - Ressurgimento da IA
  - Docker e Kubernetes
  - Pipelines de processamento distribuído (MapReduce, Spark)
  - Streaming architectures (Kafka, Pulsar)
  - Event sourcing
  - Onipresença do software livre
  - etc



# Tenha a mente aberta

- Evite pré-julgamentos
  - Gente com menos experiência pode ter soluções melhores que a sua;
  - Não seja arrogante
- Esteja aberto a mudanças (o mercado vai mudar!)
- Nenhum código está escrito em pedra
- Seja flexível;
- Não se deixe levar por estereótipos e cultos de personalidade;
- **Você não é seu código!**



# Use a ferramenta correta para o problema certo

- *“Para quem tem somente um martelo, tudo é prego”*
- A solução para um problema pode ser em uma linguagem diferente, em um framework diferente, em uma API diferente;
- “Pensar fora da caixa”;
  - Favorecida pela quantidade de conhecimento diverso que é acumulado;





# Planeje antes de codificar

- Resista ao impulso de sair codificando antes de planejar;
- Horas de design podem salvar dias e dias de codificação desnecessária;
- **O melhor código é aquele que não precisamos escrever;**
- Tenha em mente a **big picture**;





# É um jogo em equipe

- Integração
- Eficiência
- Aprendizado
- Comunicação

**Não seja arrogante  
Nem dono da verdade**



# TI é uma atividade humana

- Saiba se expressar
- Estabeleça canais de comunicação saudáveis
- Trate os outros com respeito
- Resolva conflitos



# Participe ativamente da comunidade

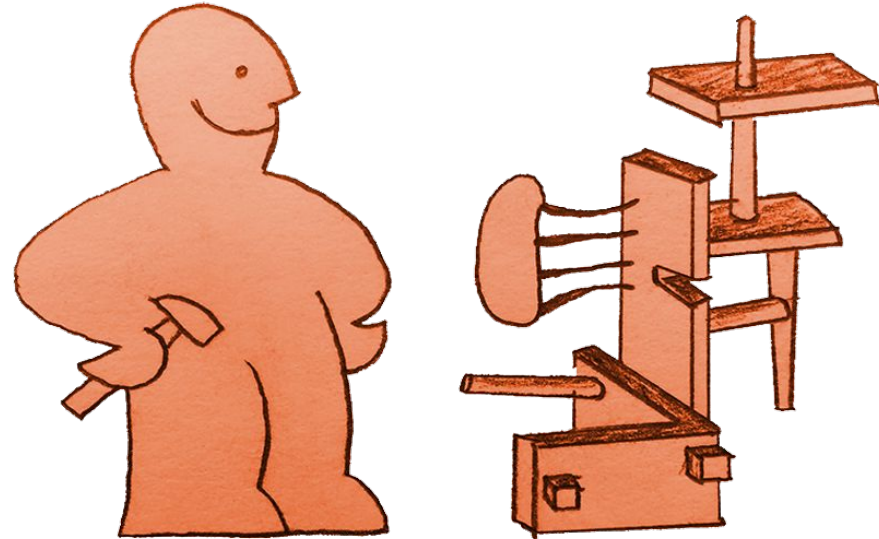
- A melhor forma de aprender é ensinando;
- Participe de:
  - Meetups
  - Congressos
  - Listas de discussão
  - Dojos de programação

***Networking, difusão de conhecimento e aprendizado, crescimento profissional.***



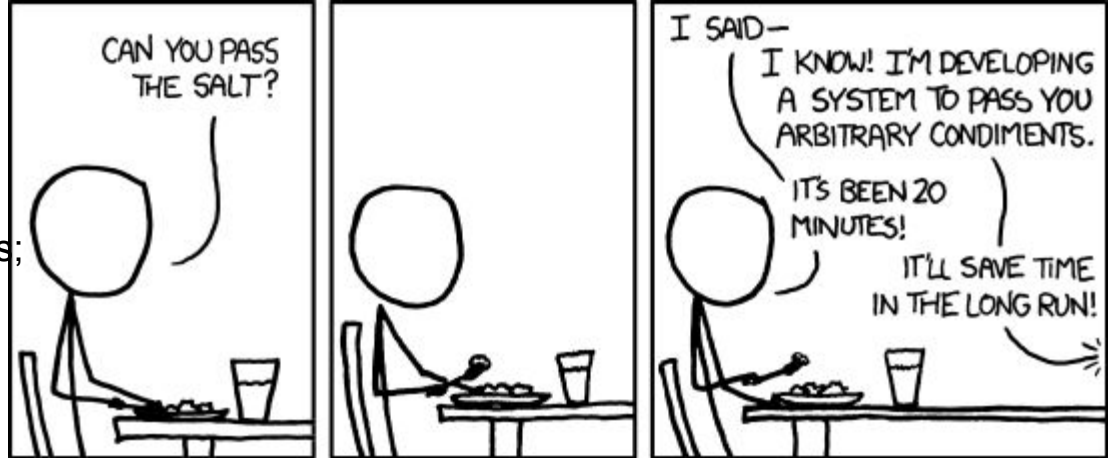
# Keep It Simple, Stupid

- O mito do programador genial
  - Sempre existirá alguém melhor, mais rápido ou mais inteligente que você (e tudo bem)!
- Tentativa de antecipar o futuro;
- Mostrar o quanto é inteligente;
- Invariavelmente leva a sistemas frágeis, **ultra-complexos** e difíceis de manter;

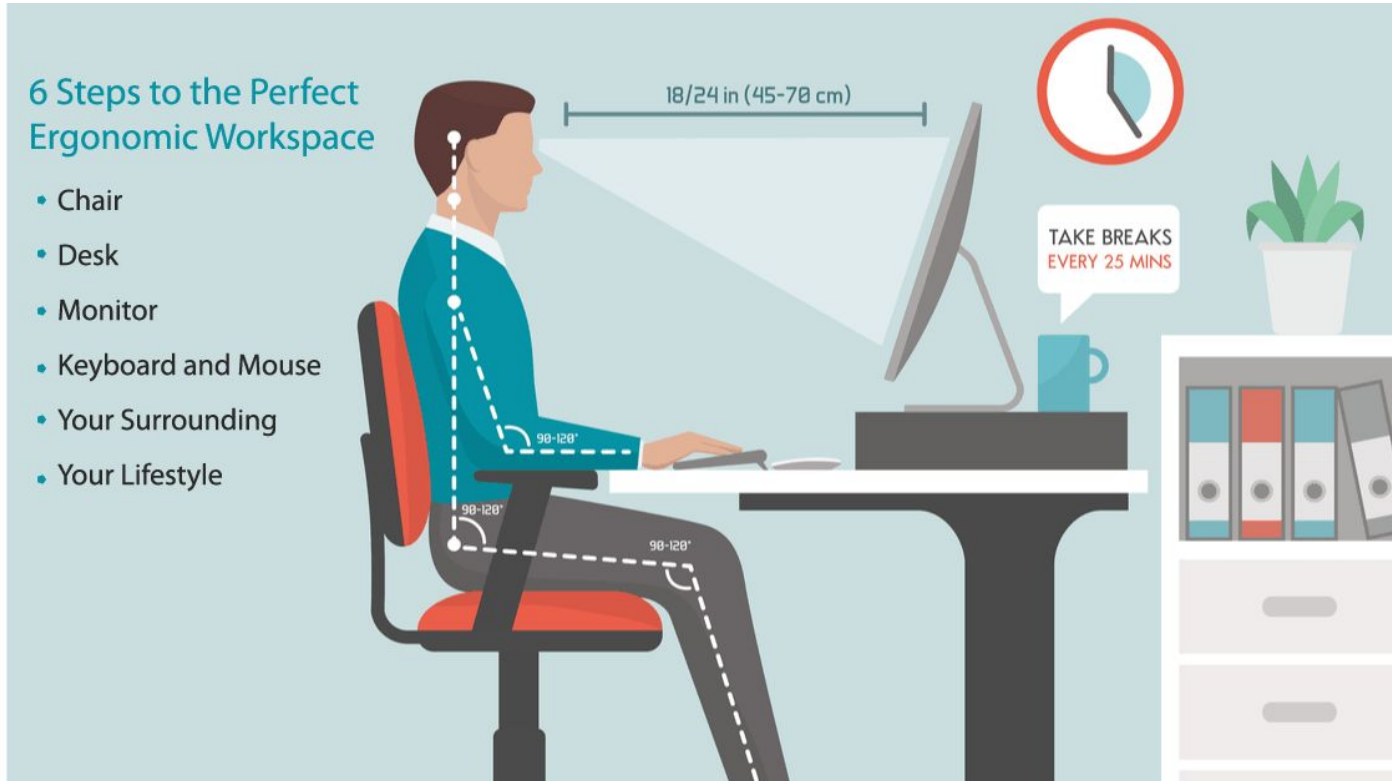


# A síndrome dos geradores de aplicação

- Degeneração do “automatize as tarefas”;
- A intenção é boa:
  - Maior produtividade;
  - Automatizar o desenvolvimento de sistemas (“fábrica de software”);
  - Desafio interessante;
- Os resultados são **péssimos**:
  - Frameworks engessados;
  - Sistemas limitados;
  - Difícil manutenção;
  - Falta de suporte;
  - Tempo e \$\$\$ desperdiçados;
  - Insatisfação da equipe;



# Ergonomia é importante!





# Cuidado com o Burnout

- **Não tente ser a heroína do projeto;**
- Não se estresse com o que não está sob seu controle;
- Encontre o balanço entre vida pessoal e profissional;
- Pratique esportes!
- Tenha hobbies que não envolvam informática!
- Tenha uma vida social (amigos, família, etc)
- E, acima de tudo,...



Não desista!



***Obrigado!***