Desenvolvimento e arquitetura de software

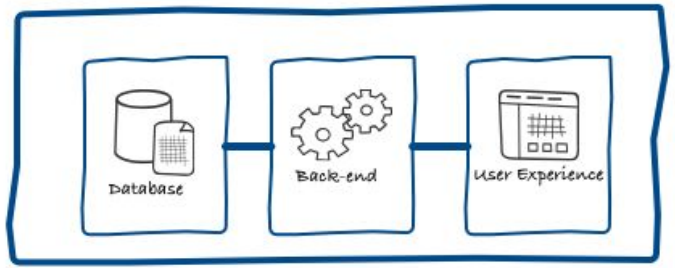
Daniela Morais
@danielammorais

$ whoami

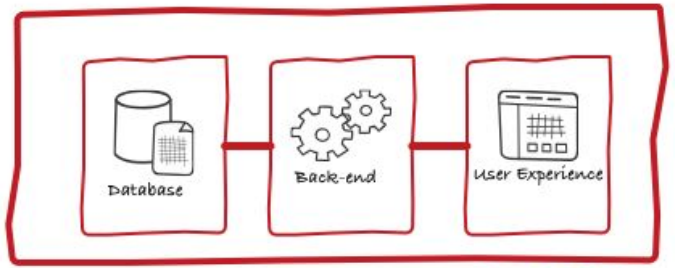
Entusiasta de Java e de códigos “limpos”
Ativista de software livre

danielammorais.com / @danielammorais

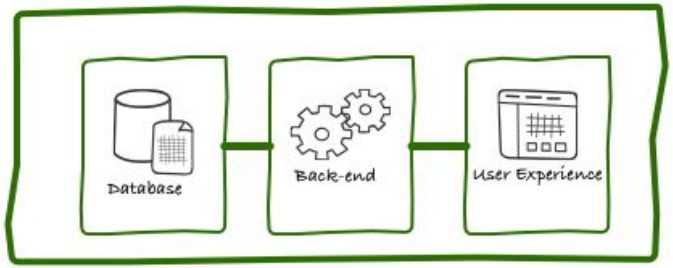
front vs back



Financial Reporting



User Manager



Point of Sale

front vs back

construção de interfaces com HTML, Javascript, CSS, React, Vue

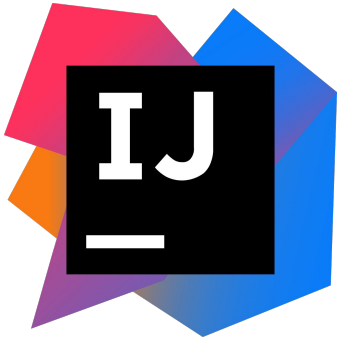
linguagens back-end: Java, Python, Ruby, PHP, GO

back-end

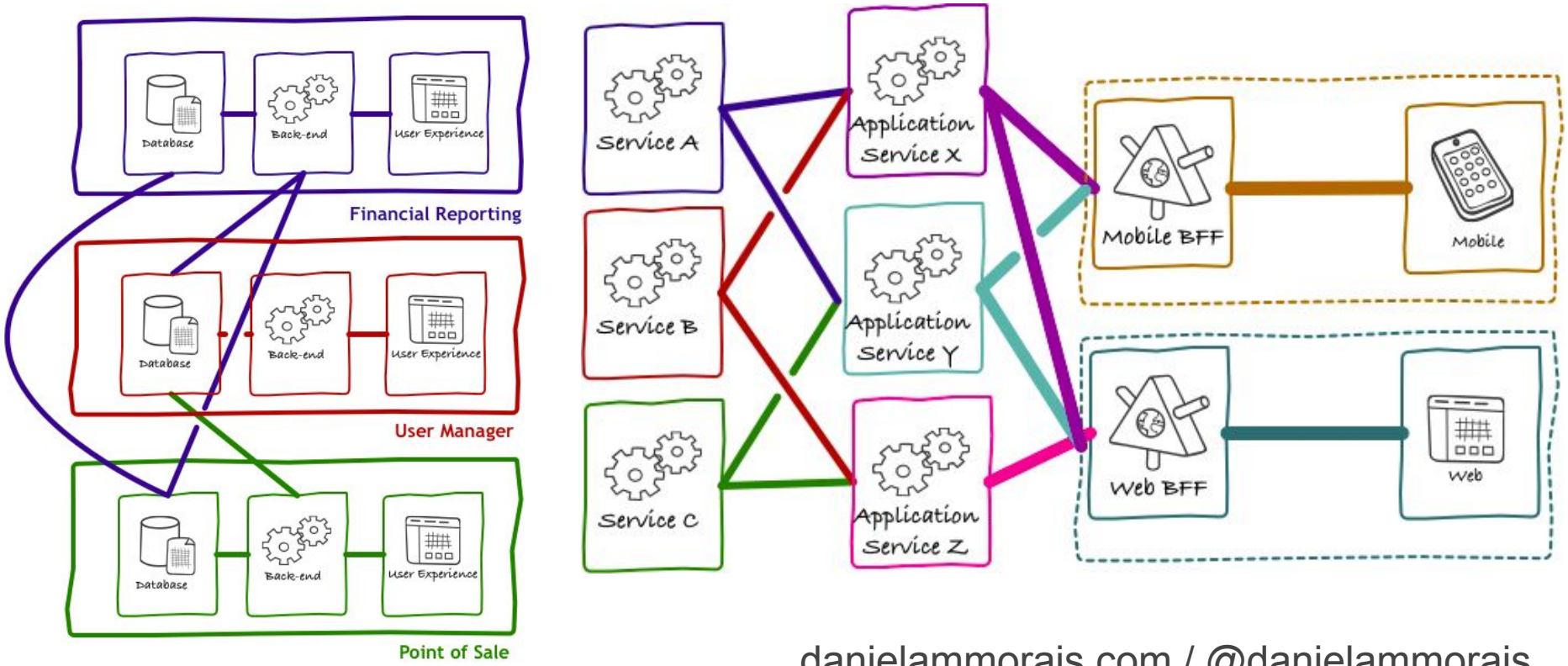
processar e manipular dados vindos do front-end

responsável por quebrar as interfaces nas férias do front

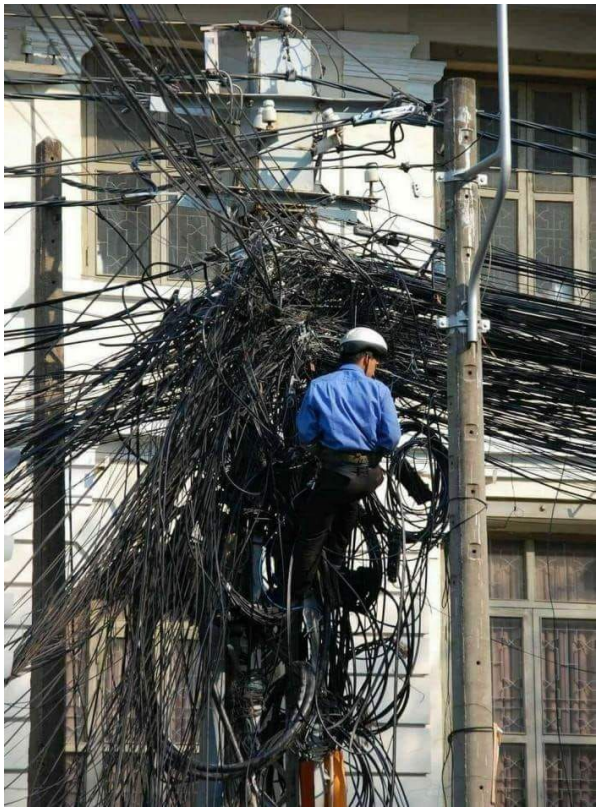
back-end



desafios


Database
Back-end
User Experience
Financial Reporting
Database
Back-end
User Experience
User Manager
Database
Back-end
User Experience
Point of Sale
Service A
Service B
Service C
Application Service X
Application Service Y
Application Service Z
Mobile BFF
Mobile
Web BFF
Web

“é só uma correção pequena, não pode ser tão difícil”



Em média 25% a 50% do esforço é gasto em entender o código

custos (Lientz e Swanson)

desenvolvimento: 43,3%

manutenção: 48,8%

outros: 7,9%

entregando sob pressão 🔥

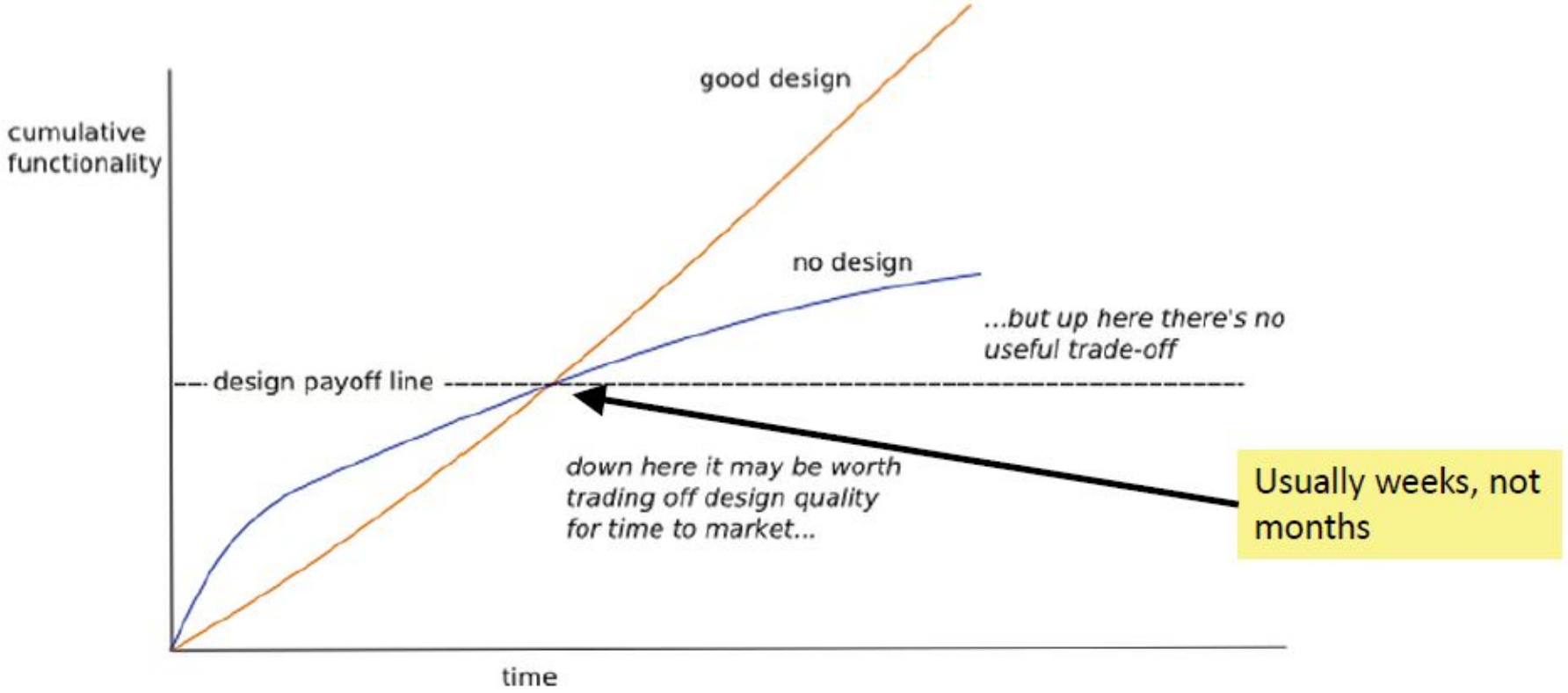
poucos testes
processos manuais
dependência de uma única pessoa
muitas features e pouco tempo
falta de conhecimento técnico/processos

“Se houver algum bug, vamos esperar que o usuário abra uma issue.”

“Não posso parar todo o time para fazer isso e não podemos contratar novos developers.”

“Product owner é o responsável por levantar os requisitos. Eu apenas codo.”

Design Stamina Hypothesis



aplicar metodologia ágil neste cenário

Você é o responsável

agile architecture

Permitir incrementar novas features

Fácil de modificar questões arquiteturais

“Build it right the first time”

Cultura ágil
Corrigir dívidas técnicas e erros nas próximas sprints
Liberdade técnica

Cultura ágil

Ambiente seguro para expor ideias
Expor visualmente todos os componentes do sistema

Cultura ágil

Revisão de código
Pair programming
Testes automatizados
Padrão de código
Clean code: code smells

code smells

A code smell is a surface indication that usually corresponds to a deeper problem in the system (Martin Fowler)

REFACTORING
·GURU·

★ Premium Course

Refactoring

Design Patterns

Hello world!

Refactoring.Guru makes it easy for you to discover everything you need to know about refactoring, design patterns, SOLID principles and other smart programming topics.

The primary purpose of this site is to show the big picture. I want to demonstrate how all these subjects intersect, how they work together and how they are still relevant. I don't pretend to be the inventor of these concepts–most of them were invented by others during the past 20 years. But I think that the connection between refactoring, patterns and general programming principles still remains a mystery for the majority of programmers. And this is the problem I would like to solve here.

*P.S. While the project is constantly being updated, you can already find tons of info on refactoring and design patterns right here on the website. The progress can be tracked via **email** or on **Facebook**.*

Alexander Shvets

The one-man band behind Refactoring.Guru

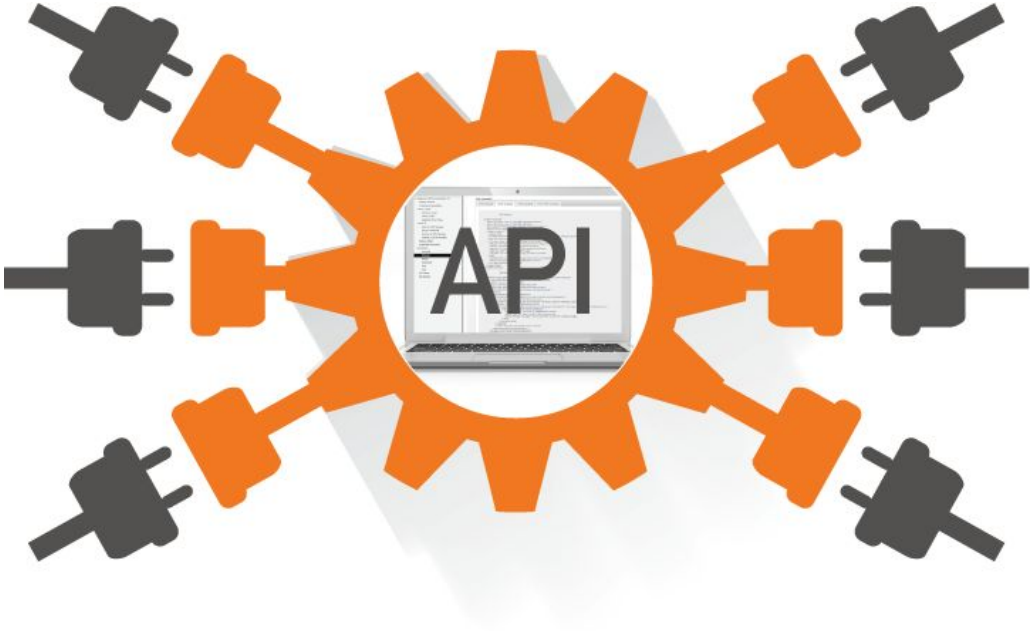
Há dois tipos de desenvolvedores ruins: Os que fazem tudo o que você diz e os que não fazem nada

danielammorais.com / @danielammorais

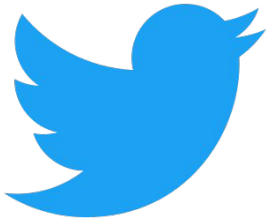
APIs

revolução digital

novos negócios



Gerar valor com APIs



danielammorais.com / @danielammorais

princípios

stateless

uso correto dos métodos HTTP (GET, PUT, PATCH, DELETE)

respostas podem ser cacheadas

princípios

GET localhost:8080/usuarios

POST localhost:8080/usuarios

PUT localhost:8080/usuarios

princípios

Não utilizar verbos em URIs

GET: Obter

PUT: Atualizar

PATCH: Atualizar parcialmente

DELETE: Deletar

princípios

Exemplo

dicas

Encontrar o paradigma que faça sentido para o negócio

Lidar com a pressão e blindar o time

Pessoa aberta a novas ideias e sugestões

software & comunidade

“A economia do século 21 não é sustentada por aço.
A economia do século 21 é sustentada por software.
Software é elemento tão fundamental para o desenvolvimento econômico no século 21 quanto o foi a produção de aço no século 20. (...)
E a boa notícia é que ninguém o possui.”
Software e Comunidade no começo do Século 21

Créditos

Making Architecture Matter - Martin Fowler
Design Stamina Hypothesis:
https://martinfowler.com/bliki/DesignStaminaHypothesis.html