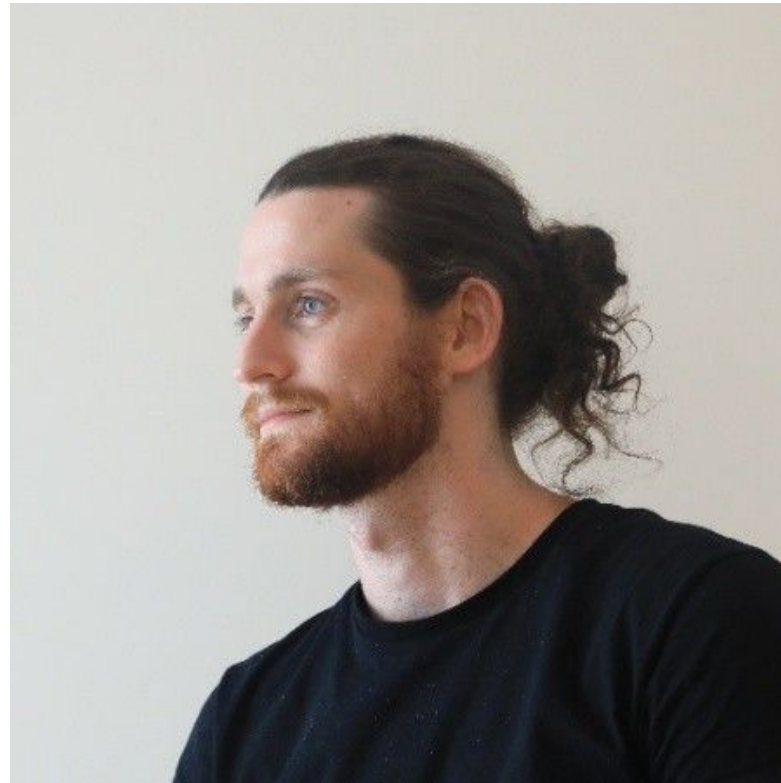


Colocando Ideias em Produção

Ariel Schvartz





Ariel Schvartz

[linkedin.com/in/arielschvartz](https://www.linkedin.com/in/arielschvartz)

Agenda



Design Thinking

Entendendo a ideia e mapeando a solução

Design

Materializando a ideia e colocando ela no papel

Definindo a Arquitetura

Modelo de dados, comunicação e etc

Escolha de Linguagens e Frameworks

Vantagens, desvantagens e pontos de decisão

Desenvolvimento de Código

Boas práticas, agilidade e eficiência.



Testes

Ambientes de testes e testes automáticos

CI / CD

Automatizando o processo de colocar em produção

Escala

Como arquitetar e otimizar o servidor para aguentar carga

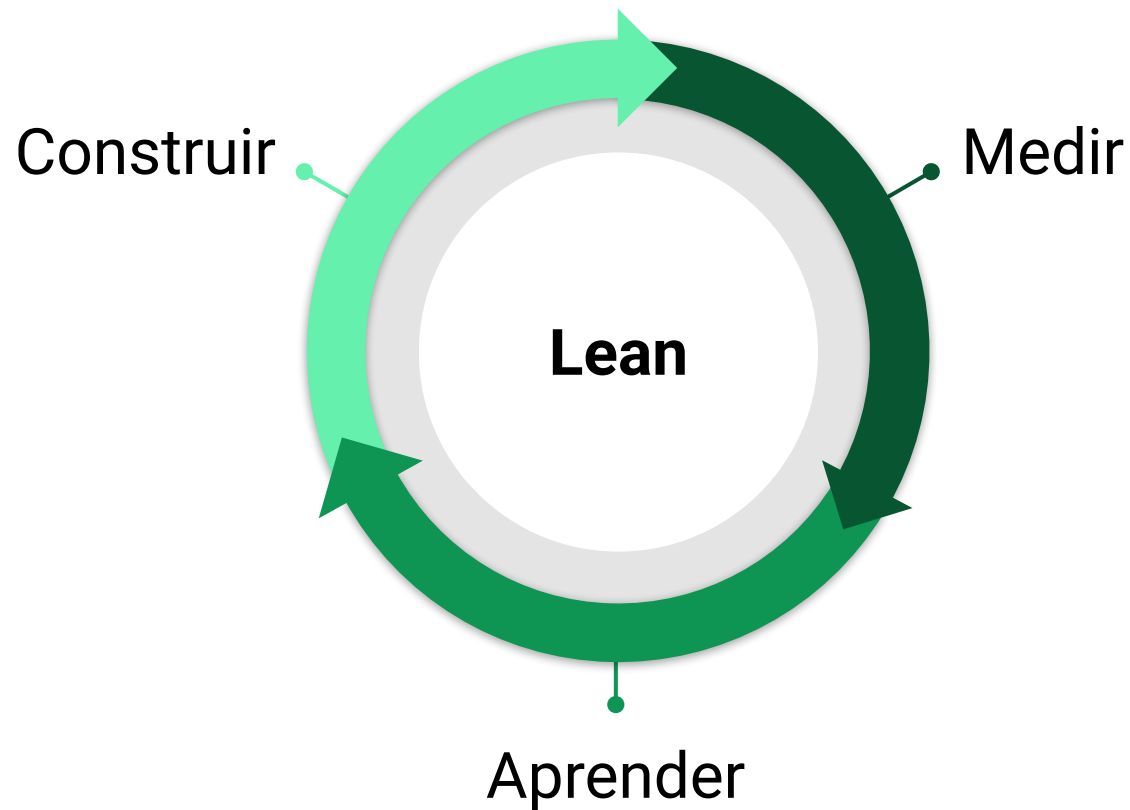
Casos Reais

Lawgile
Rupee
Medicamentos Inteligentes
BGC Brasil

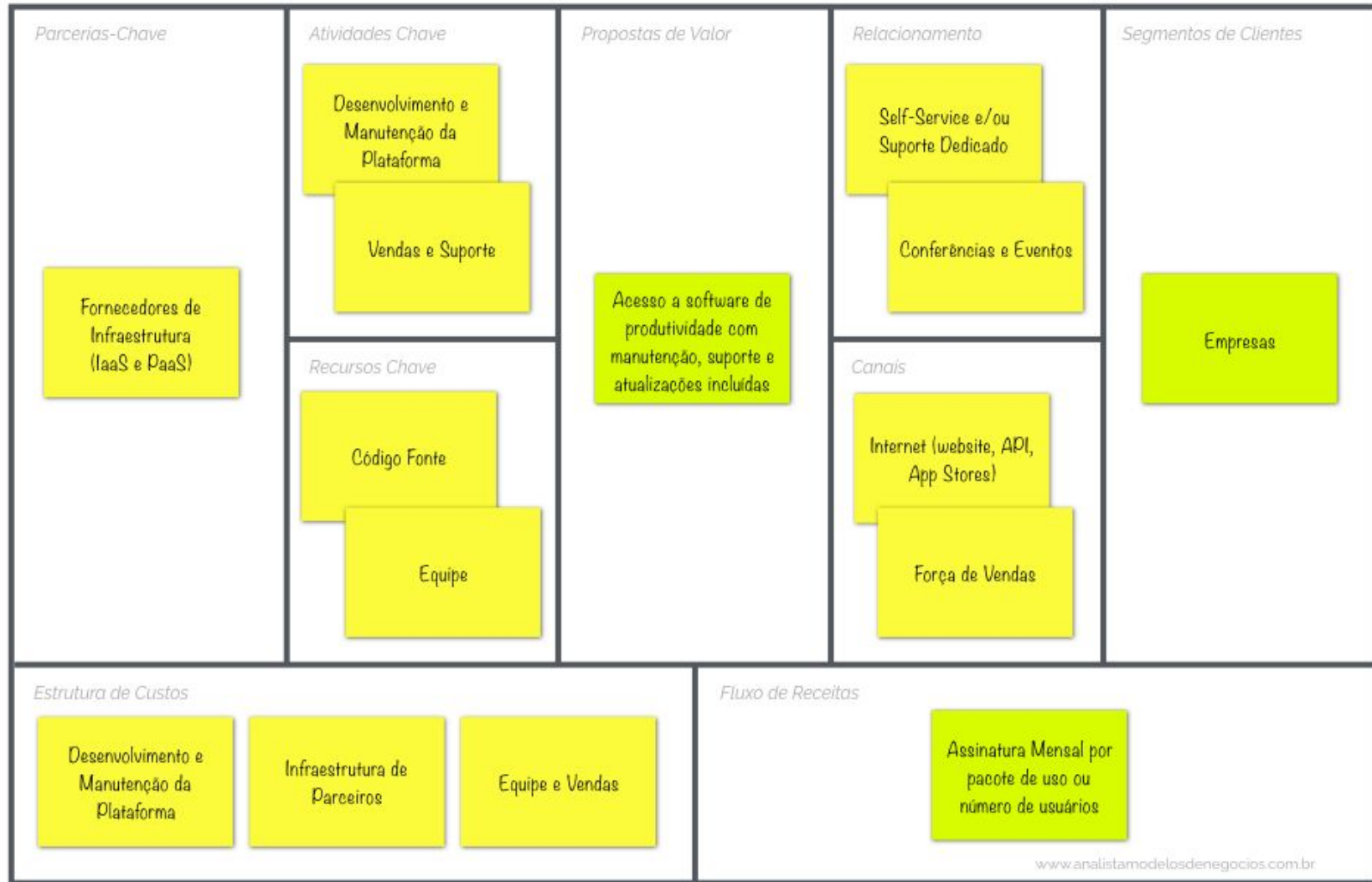
Serverless

Conceito e como usar o Framework

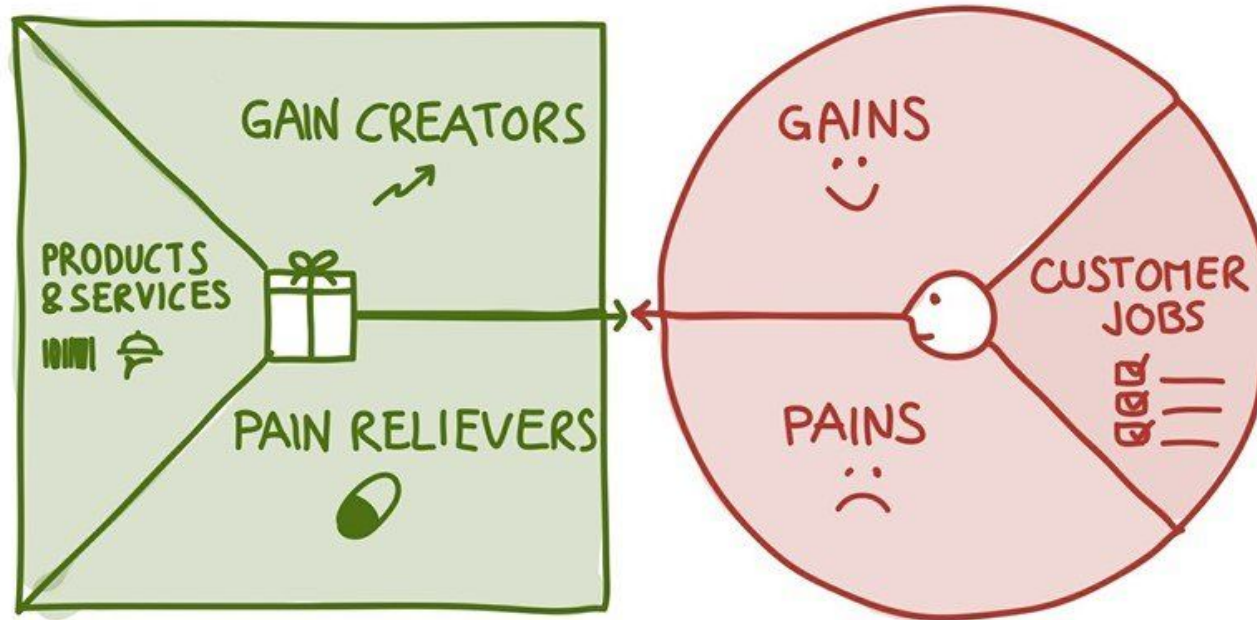
Lean Startup



Business Model Canvas



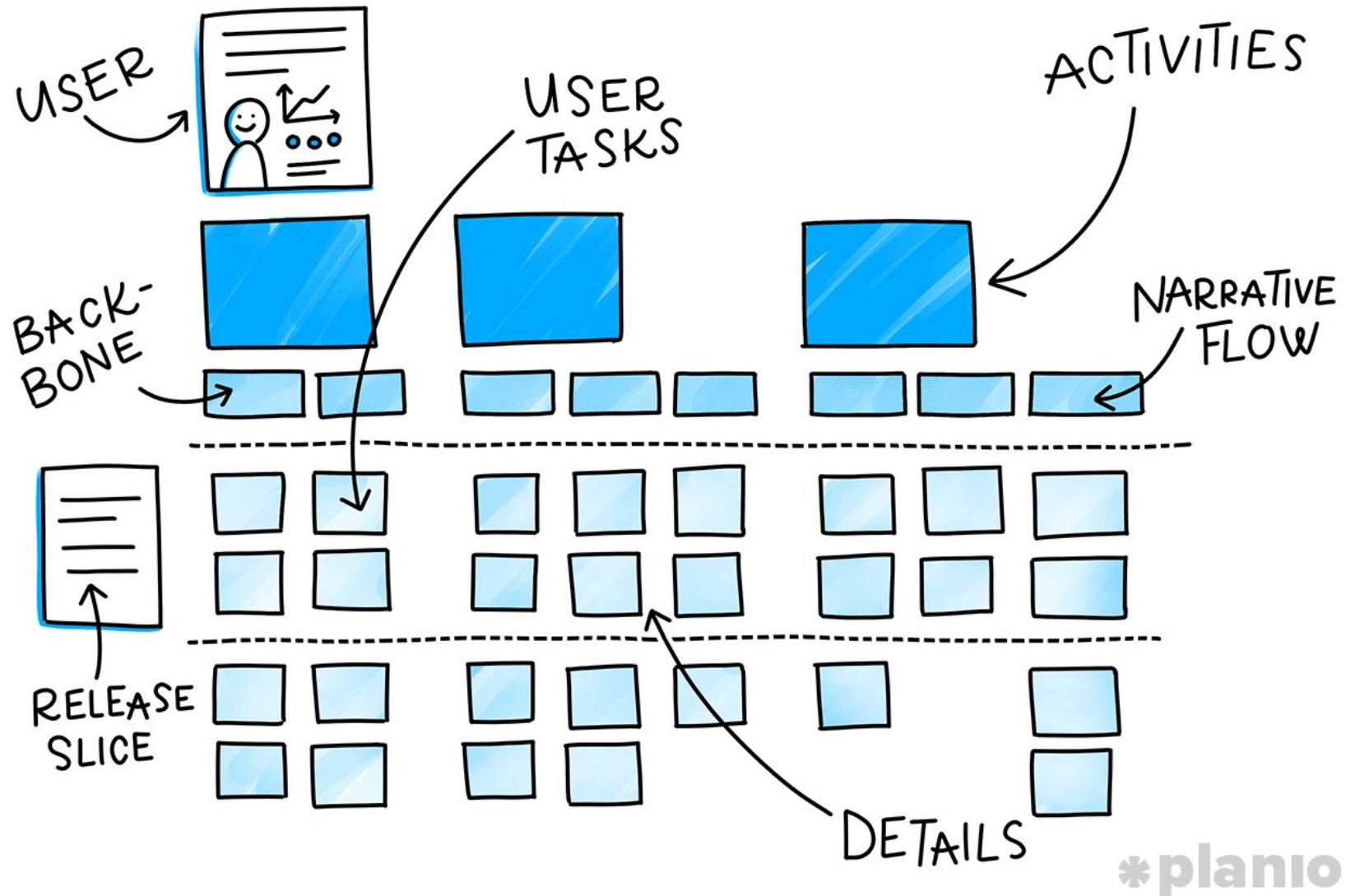
Value Proposition Canvas



Design Studio

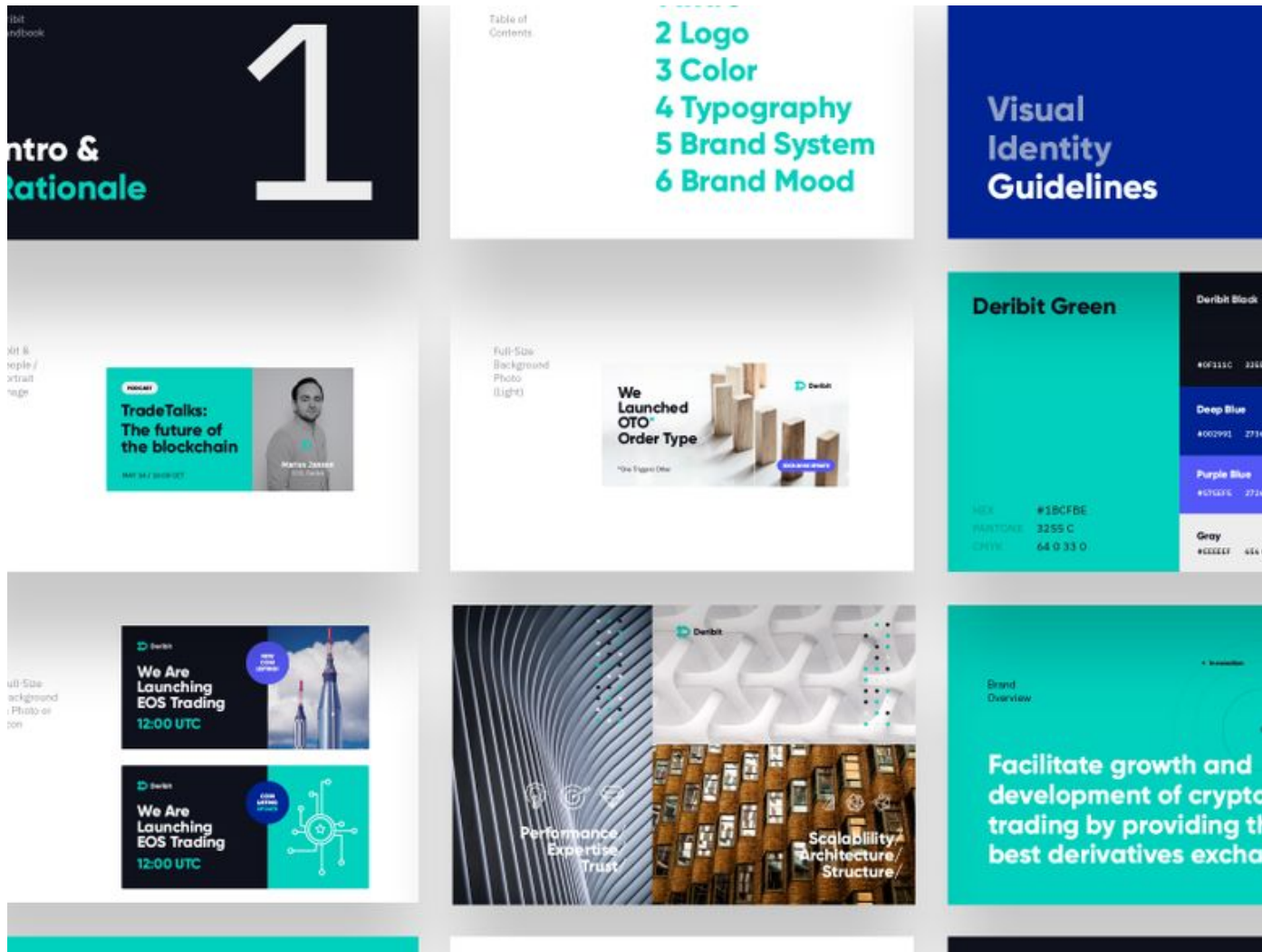


User Story Mapping



*planio

Identidade Visual



Wireframes



Layout



Culture



Denis Leary's No Cure for Cancer / Still electrifying and obnoxious 25 years on

143



Scotland / Top 10 books about the Highlands and Islands

30



From Hard Sun to Lost / Why I can't stop cheering on TV's bad guys

122



The Secret Twenties by Timothy Phillips review / Spies and the Bolshevik threat

5

+ More Culture



Susan Meiselas / From the dressing room to the danger zone - in pictures



Photography Susan Meiselas on how she shot Molotov men, war widows and carnival strippers



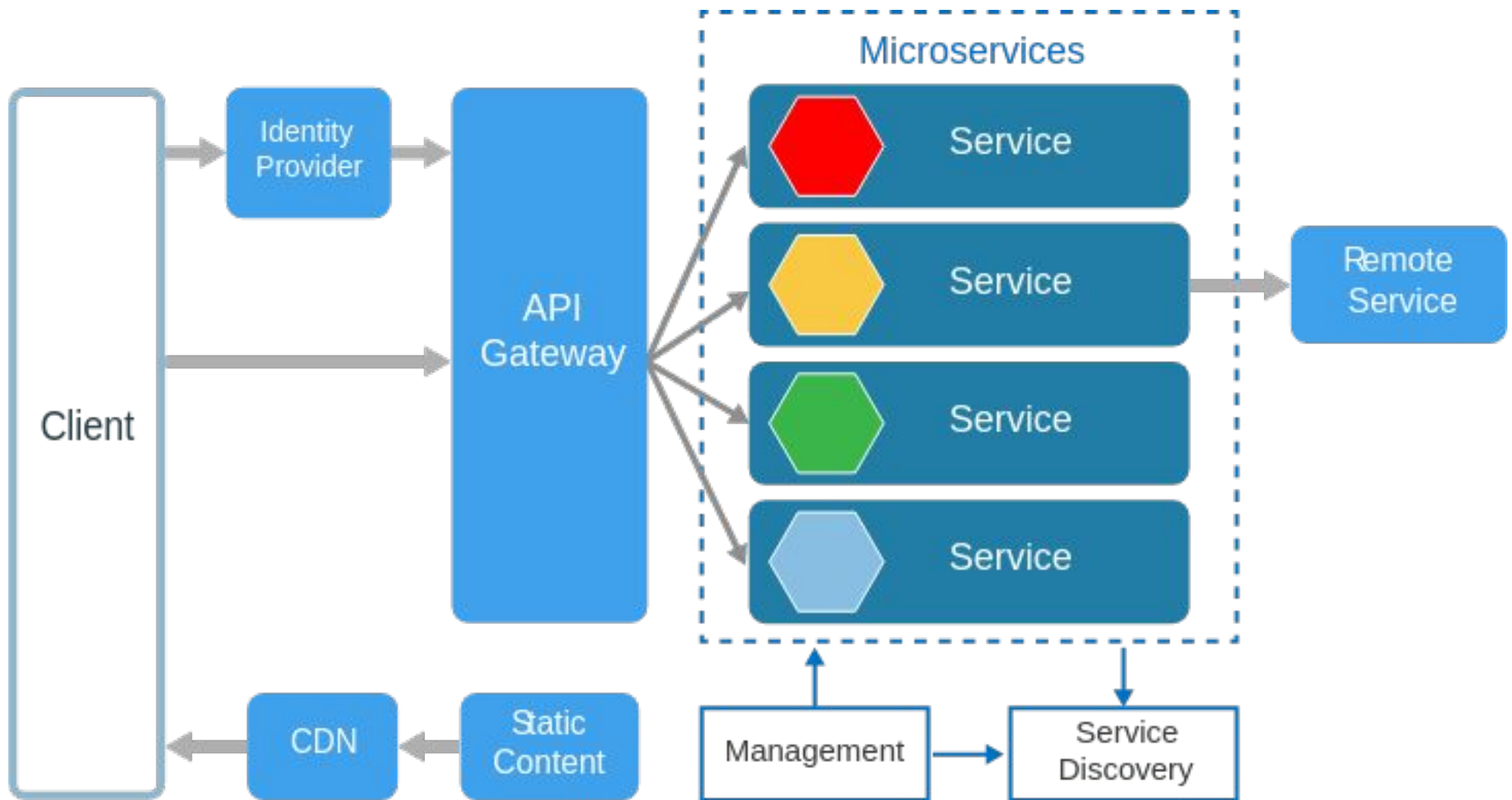
Queer Eye review / Netflix reboot looks beyond the wardrobe to find hidden depths

Sexuality is not the main issue here. Instead, lifestyles and prejudices are challenged with a frank and heart-warming honesty that is hugely affecting

8

Hide

Arquitetura do Sistema



Partes do Sistema

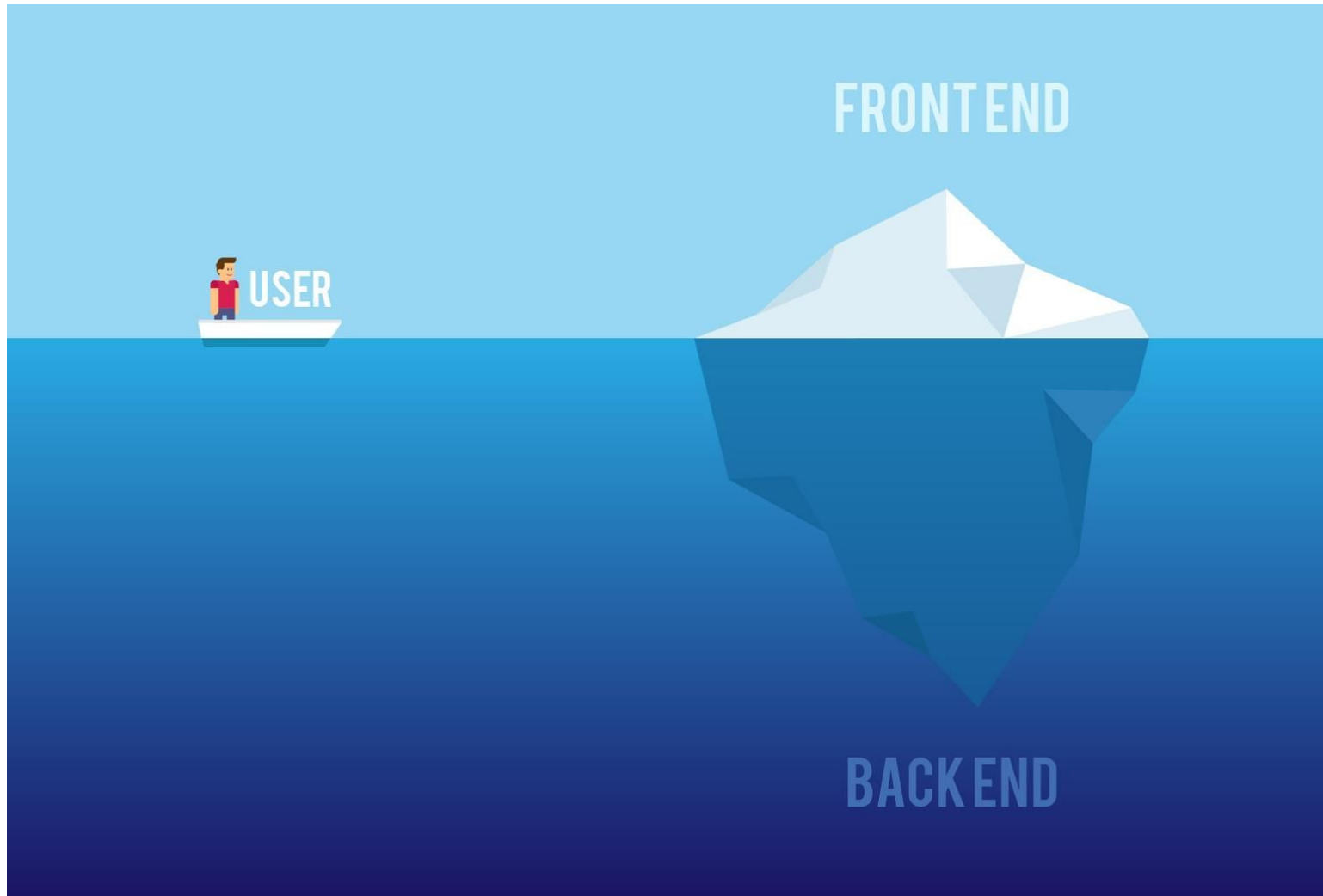
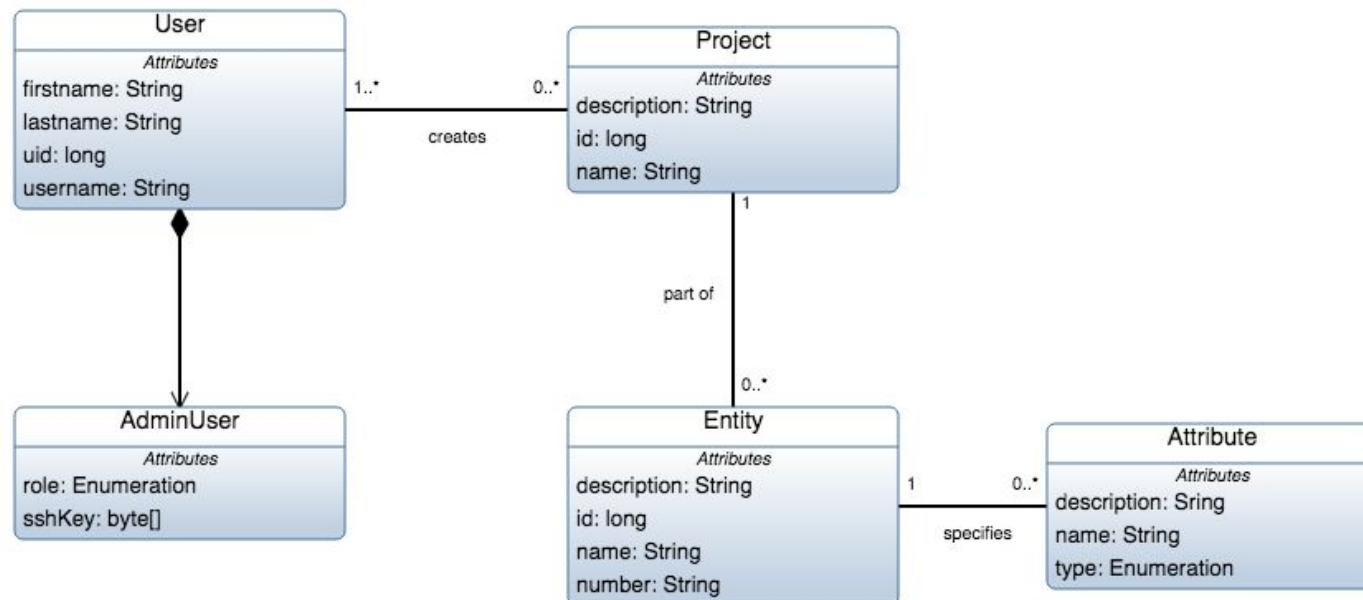


Diagrama de Classes

- Como vamos organizar o Banco de Dados?
- O que precisamos guardar?
- Quais entidades existem no sistema?
- Como as entidades se relacionam?



Arquitetura do Banco de Dados

- Centralizado
- Distribuído
- Replicado
- Blockchain
- Bancos Relacionais (SQL)
 - MySQL
 - PostgreSQL
 - SQLite
- Bancos não Relacionais (NoSQL)
 - Chave-valor
 - Documento
 - GraphDB (BD orientado a grafos)
 - Colunas

Arquitetura do Banco de Dados

SQL

- Modelagem pré-definida
- Consistência de Dados
- Bom para queries complexas de dados
- Costuma precisar de uma infra mais cara

NoSQL

- Dinâmico
 - Não precisa pré definir colunas
- Performance
 - Consultas simples
 - Escrita
 - Escala horizontal

Estrutura do Código

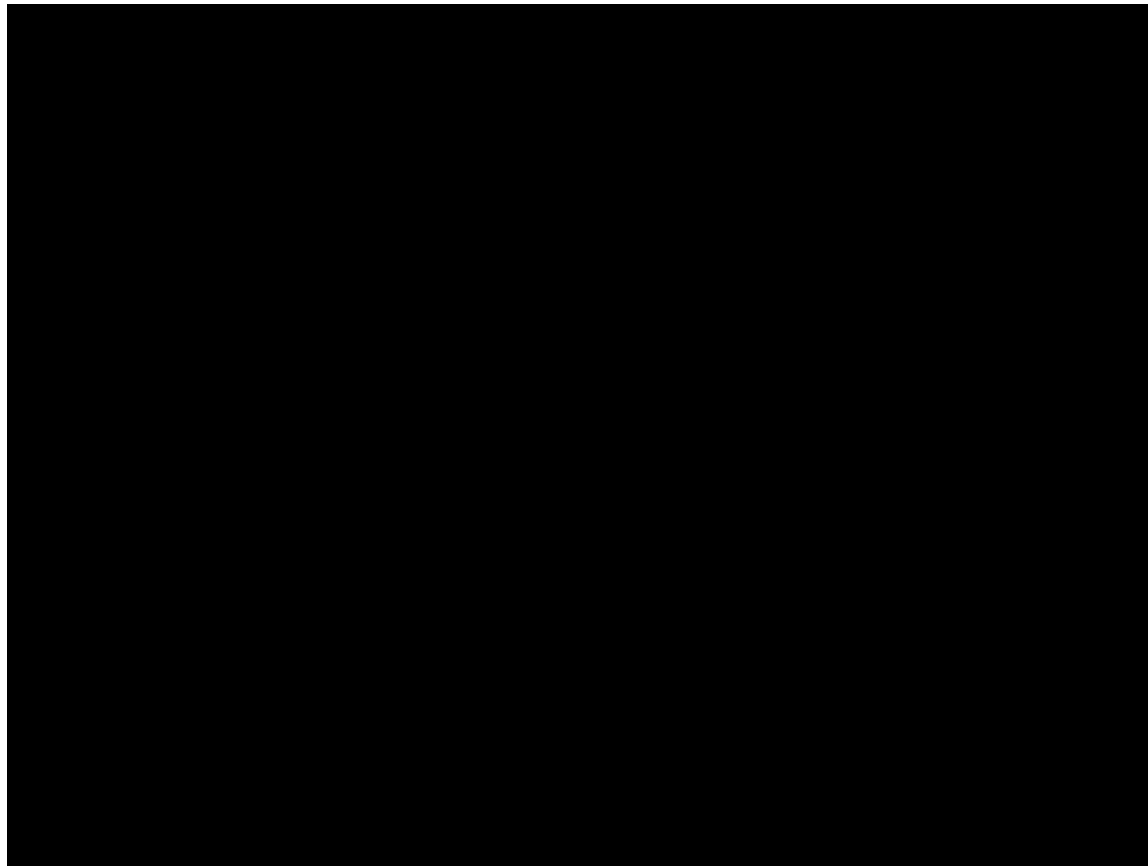
Monolito

- Tudo num mesmo projeto
- Deploy único
- Gestão única
- Código mais amarrado e mais difícil de manter
- Mais ágil pro MVP

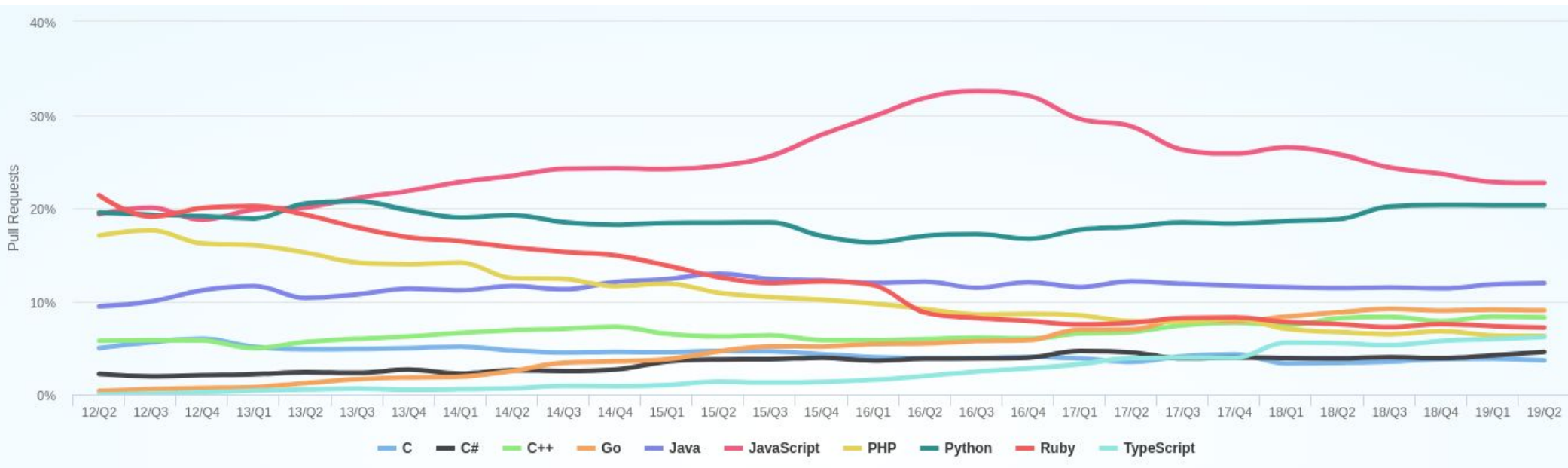
Micro serviços

- Comunicação por APIs
- Bancos de Dados independentes
 - Replicação do necessário
- Múltiplos deploys
- Multi linguagens
- Otimização por serviço

Qual linguagem escolher?



Qual linguagem escolher?



Qual linguagem / framework escolher?

- Quais as vantagens específicas da linguagem?
 - Velocidade de Processamento? Agilidade no Desenvolvimento?
- Qual o canal de interação do usuário?
 - Internet vs Tempo de Processamento
- Qual o tamanho da comunidade?
 - A comunidade costuma usar com o mesmo objetivo?
 - Tem bastante lib open source?
 - É fácil achar gente no mercado de trabalho?
- Meu case me obriga a algum caso específico?

Qual linguagem / framework escolher?

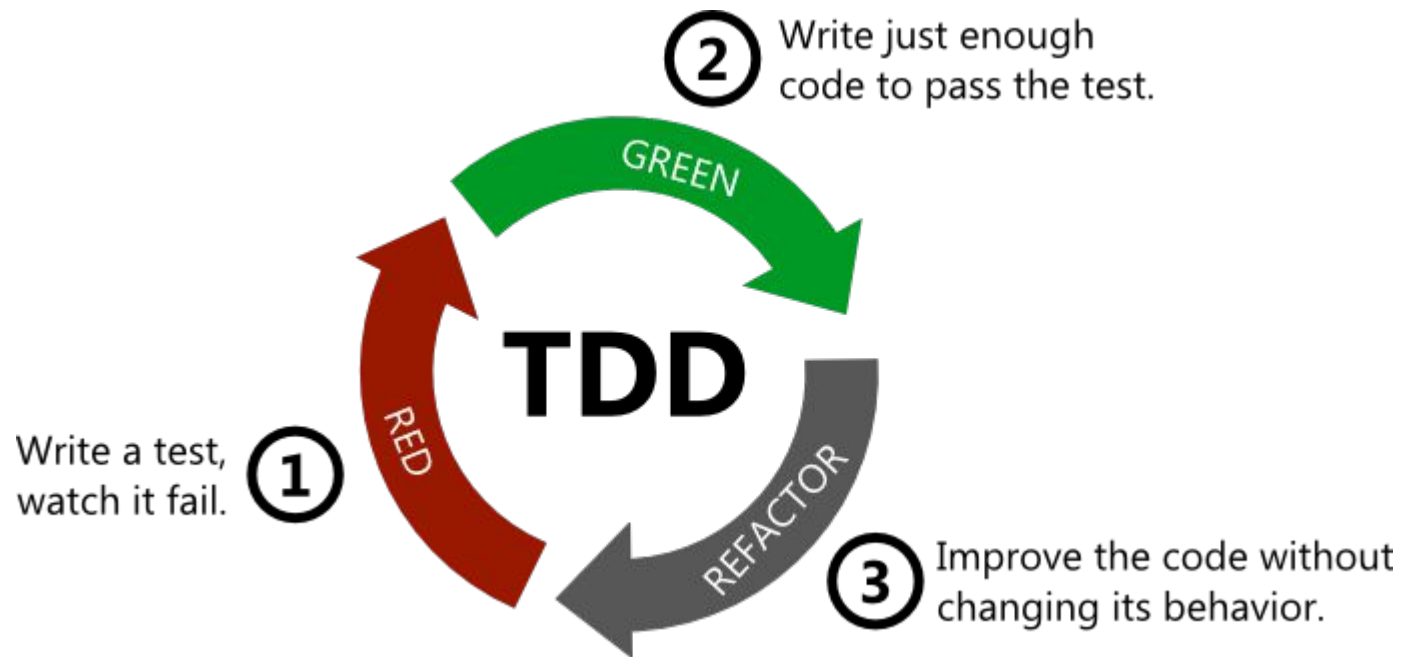
- AirBnB
 - Rails + React, React Native (Depois Java e Swift)
- Instagram
 - Django + React Native
- Nubank
 - Clojure, Python + React Native, Swift, Java
 - Micro serviços
- Uber
 - Python, Node (Depois Java, Go) + Node, Swift, Java
 - Micro serviços (SOA)

Desenvolvendo o código

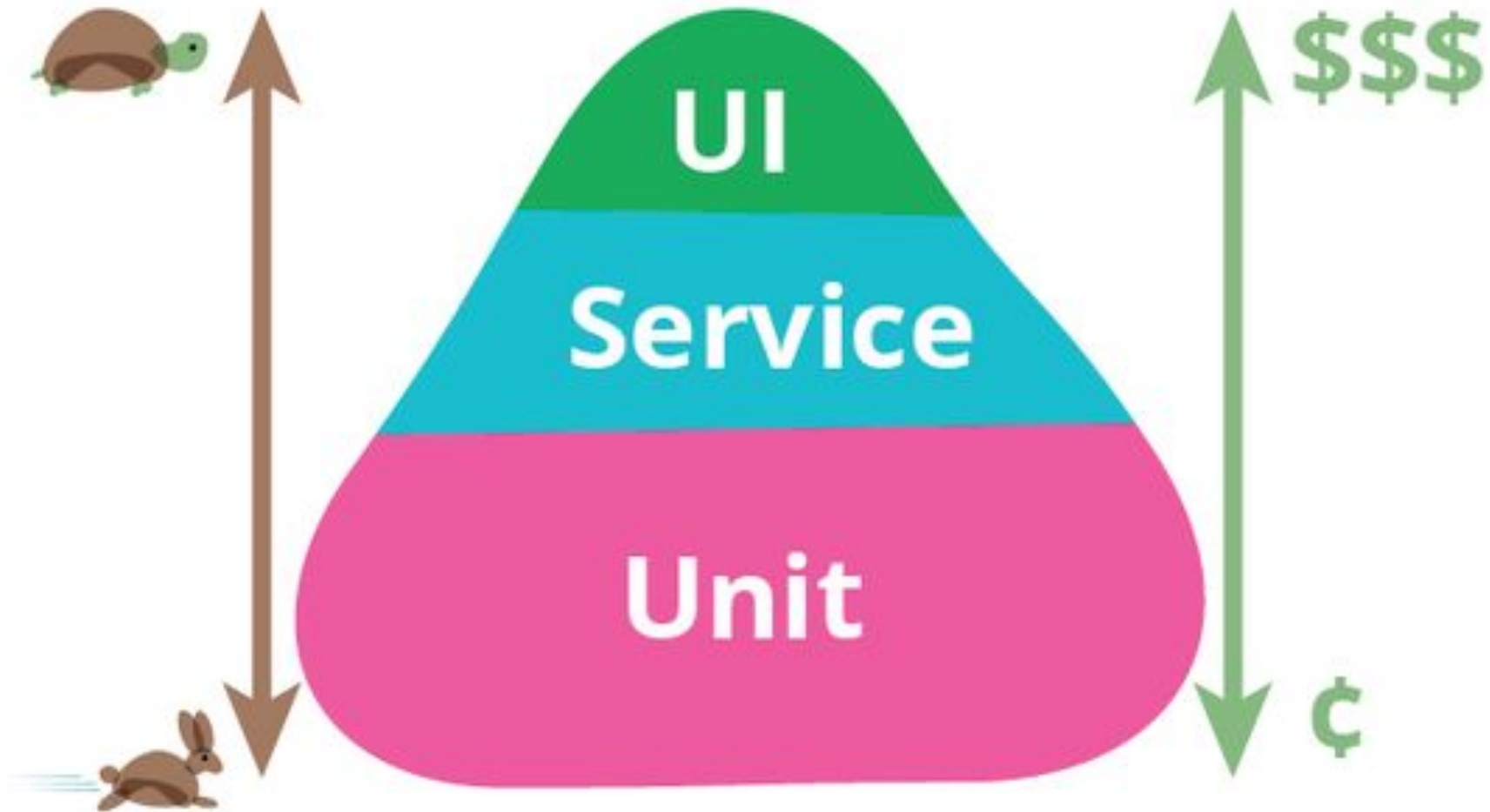
- Discutir a solução antes de começar
- Pair Programming
- Refactor Constante
- Boas práticas
 - DRY
 - Nomenclatura
- Manutenibilidade
 - COC (Convention Over Configuration)
- TDD

TDD

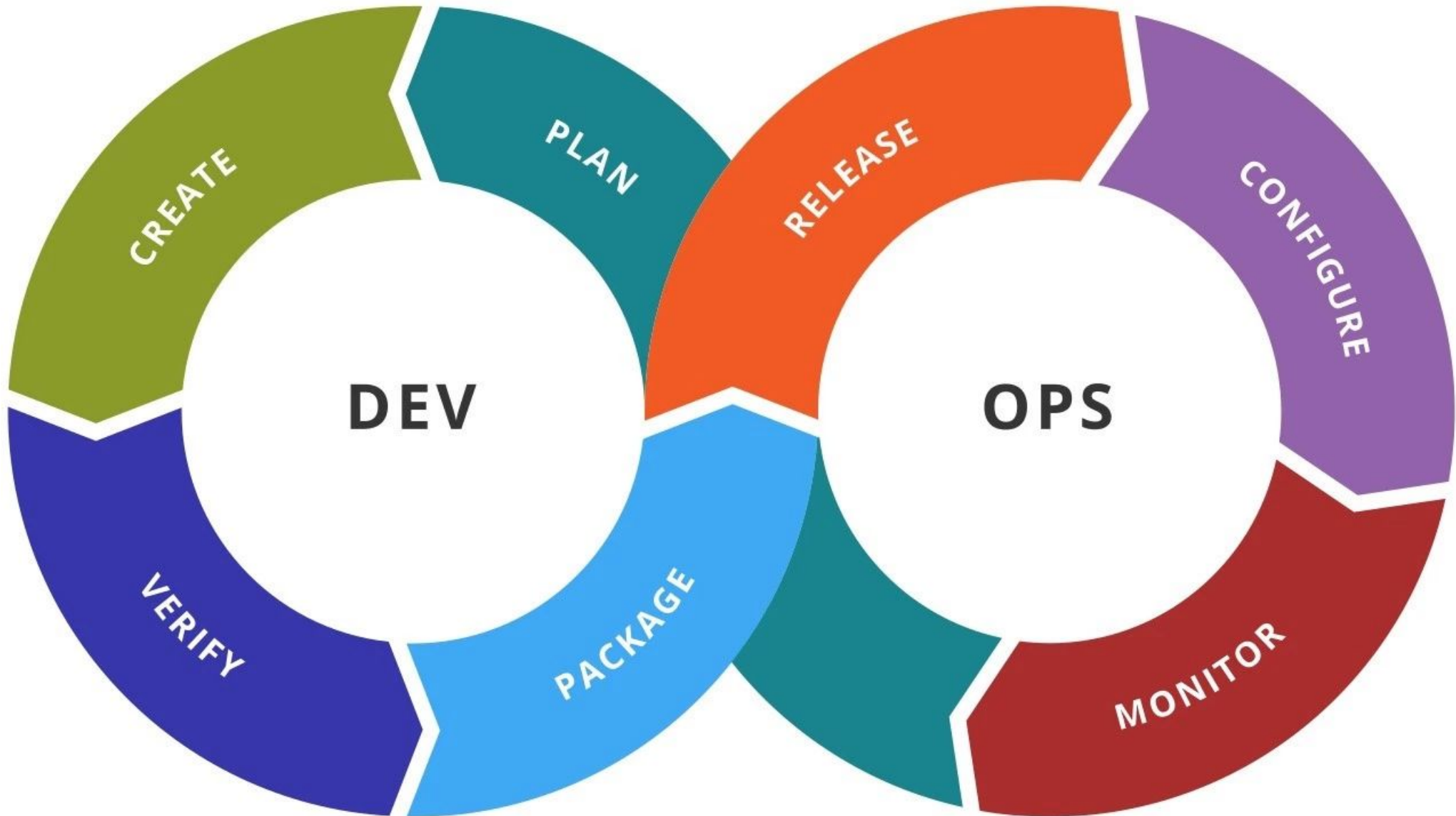
- Tende a ter menos bugs
- Ajuda a pensar em casos de uso não pensados antes
- Costuma economizar tempo



Testes Automáticos



CI / CD



Colocando em Produção

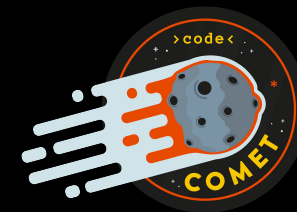
- Configurando Servidores
 - Máquina Virtual? Kubernetes? Serverless?
 - Bancos de Dados
 - Migrations & Seeds
 - Backups
- Interfaces Web
 - Sites Estáticos vs Server Side Rendering
- Infrastructure as Code
 - Terraform, Cloudformation

Escala do Servidor

- Métricas de Performance
 - Alarmes para problemas
- Logs bem estruturados
- Custos gerados
 - Como otimizar os custos?
- Downtime para mudanças
- Tempo de Resposta aceitável
- Distribuição de carga de acesso
- Estratégias de Deploy
 - Big Bang, A/B, Canary



INVENTOS digitais

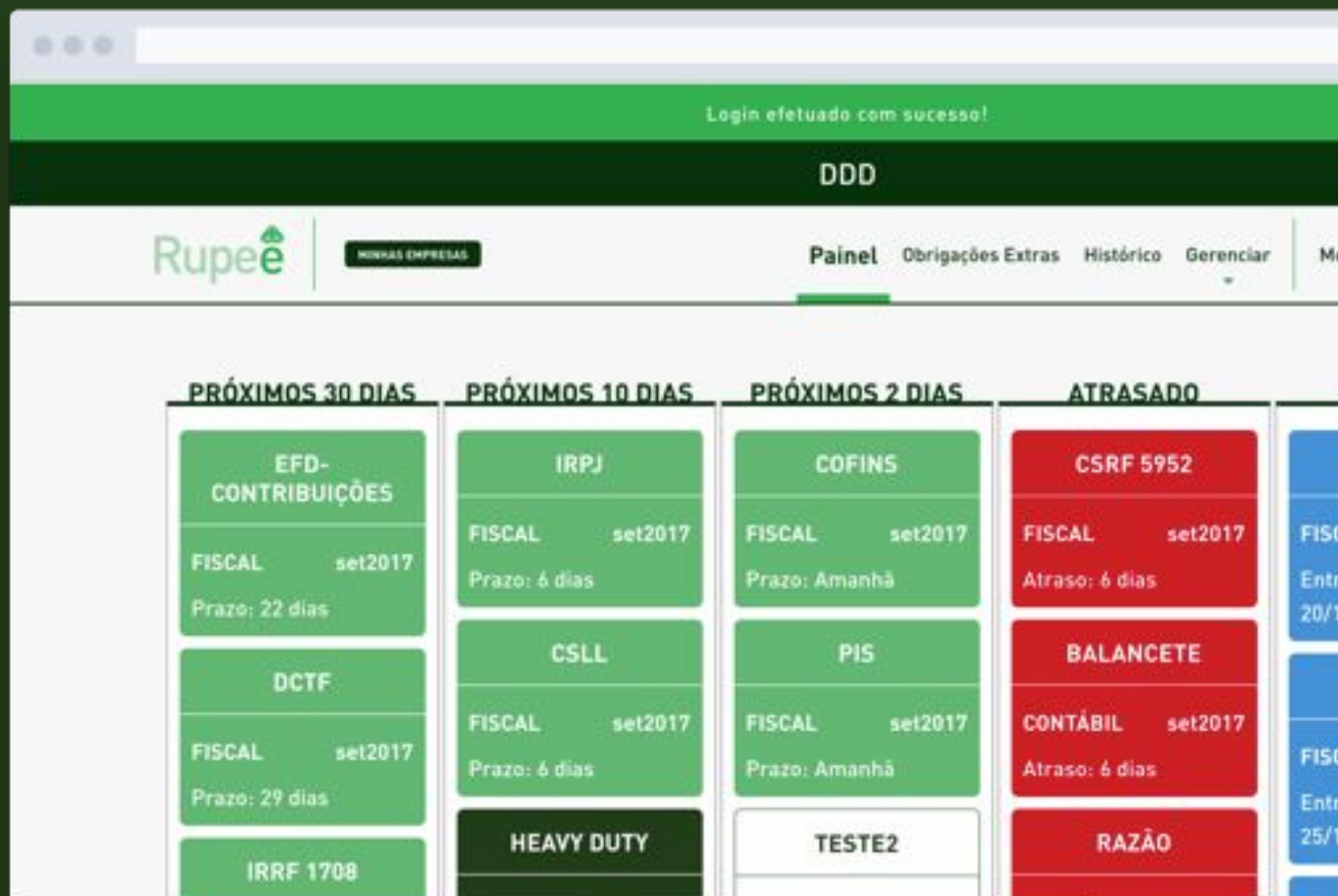




Plataforma de automação e controle contábil, fiscal e tributário.

Atendendo a:

- Grandes conglomerados
- Multinacionais
- Empresas listadas em bolsa
- Grandes franquias
- Um dos maiores escritórios de contabilidade do Brasil



Rupee

- Projeto em Rails
- Velocidade de Desenvolvimento de Produto
- Banco de Dados SQL
 - Queries complexas
 - Cache
- Segurança da Informação
 - Controle de acessos
 - Links temporários



**Eficiência e
transparência no
mundo jurídico**

Lawgile

- Projeto Serverless
- NodeJS
- React + React Native
- API GraphQL
- Processo bem elaborado de CI/CD
 - Muito integrado às validações
- Segurança da Informação

Medicamentos Inteligentes

Tecnologias para a saúde do futuro

Medicamentos Inteligentes

- Interface via Whatsapp
 - WhatsApp Business API
 - DialogFlow do Google
 - Arvore de decisão
- Backend Serverless
- Machine Learning



A bgc é a maior especialista em background checks do Brasil, com uma solução que alia tecnologia e expertise jurídica e entrega resultados rápidos e seguros para empresas de todos os tamanhos.

Saiba quem você está contratando

A bgc é a maior especialista em background checks do Brasil, com uma solução que alia tecnologia e expertise jurídica e entrega resultados rápidos e seguros para empresas de todos os tamanhos.

Soluções

- ✓ **Validação de documentos**
Extração de dados via OCR e verificação em órgãos relevantes.
- ✓ **Checagem de antecedentes criminais**
Verificação e análise de processos criminais e emissão de certidões em todo âmbito nacional.
- ✓ **Reconhecimento facial**
Comparação da foto do candidato com o documento, através de inteligência artificial e machine learning.
- ✓ **Testes de segurança**
Aplicação de exames toxicológicos e avaliações psicológicas com foco em indícios de comportamentos indesejados.

BGC

- Arquitetura Serverless
- Quebrado em micro serviços
 - BDs próprios
 - Streaming de dados
- Filas para controle de volume
- Máquinas de Estado
- Bancos SQL e NoSQL
- Otimização de custos

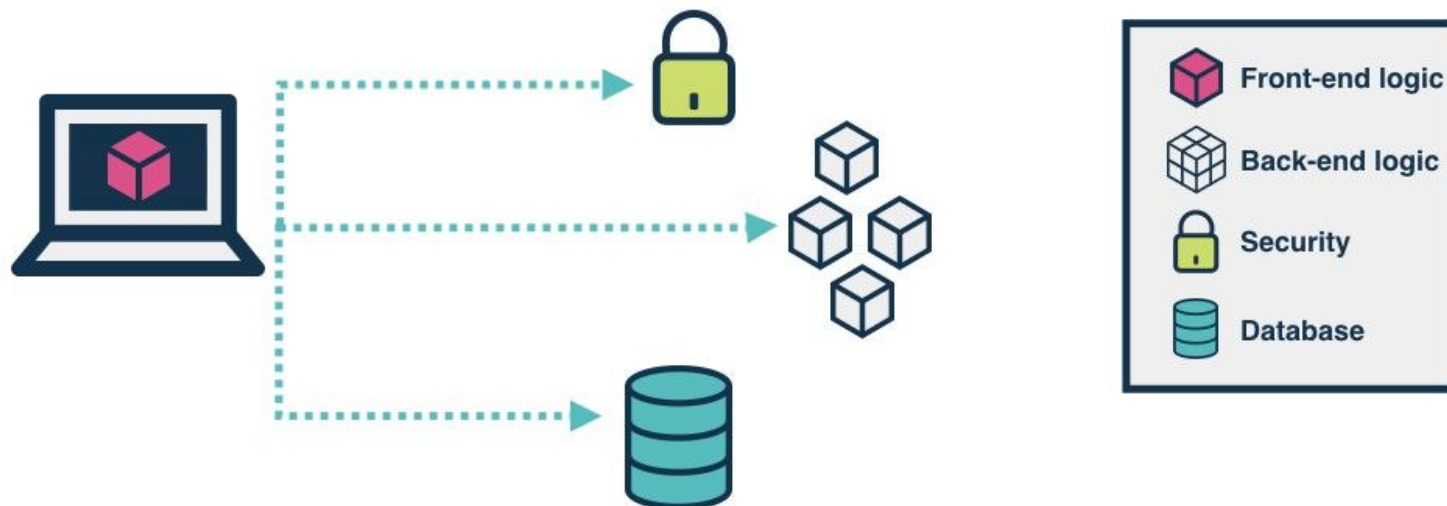
Serverless

TRADITIONAL vs SERVERLESS

TRADITIONAL



SERVERLESS (using client-side logic and third-party services)



Serverless

Vantagens:

- Zero preocupação com manutenção de servidor
- Sem custo de tempo ocioso
- Sem problemas com escala
- Atualização e deploy rápido

Desvantagens:

- Difícil de testar e debuggar
- Processos longos
- Ficar preso a provedores de infra
- Poucas pessoas no mercado com experiência

Serverless

Principais fornecedores:

- Amazon AWS Lambda
 - Mais antigo e maduro
 - Introduzido em Nov 2014
- Google Cloud Functions
- Azure Functions
- IBM OpenWhisk

AWS:

- Lambda
- Step Functions
- DynamoDB on Demand
- RDS Aurora
- Kinesis Streaming
- CloudFormation

Serverless Framework

- Organização de código
- Plugins
- Linha de comando
 - Deploy
 - Logs
- Multi Provedor



Vamos ver na prática?

File Edit Selection Find View Goto Tools Project Preferences Help

```
FOLDERS
  bgc-crawler
  active-drivers
  awsEvents
  backups
  driversMock
  examples
  resources
  scripts
  src
  models
  scrapers
  slack
  utils
  /* admin.js
  /* api.js
  /* cabifyBlacklist.js
  /* clients.js
  /* crawler.js
  /* crawlerManagement.js
  /* drivers.js
  /* master.js
  /* proxies.js
  /* recaptcha.js
  /* slack.js
  /* sources.js
  stepFunctions
  test
  .babelrc
  .directory
  /* .eslintrc.js
  .gitignore
  /* apiListDriversMock.json
  /* bitbucket-pipelines.yml
  /* driver.json
  /* event.json
  /* handler.js
  /* package-lock.json
  /* package.json
  <> README.md
  /* reprocessed.json
  /* serverless.yml
  temp.txt
  /* test_sources.json
  /* updateAllDrivers.js

clients.js
1  const { Client } = require('./models');
2  const { apiResponse, authorizeUser } = require('./utils/api');
3
4  const AWS = require('aws-sdk');
5  const lambda = new AWS.Lambda();
6
7  const {
8    updateClientStepFunction,
9    updateBranchStepFunction,
10   } = require('./utils/aws');
11
12  // Qual é o trigger para New?
13  module.exports.newClient = (event, context, callback) => {
14    try {
15      const body = JSON.parse(event.body);
16      const client = new Client(body);
17
18      client.save().then(() => {
19        callback(null, apiResponse(201, {
20          body: {
21            message: 'Client registered successfully!',
22            client: client.toJson(),
23          },
24        }));
25      }) catch((errors) => {
26        callback(null, apiResponse(400, {
27          body: {
28            errors,
29            client: client.toJson(),
30          },
31        }));
32      });
33    } catch(e) {
34      callback(null, apiResponse(400, {
35        body: {
36          message: 'invalid body',
37        },
38      }));
39    }
40  }
41
42  // Qual é o trigger para Update
43  module.exports.updateClient = (event, context, callback) => {
44    try {
45      const body = JSON.parse(event.body);
46
47      // ID obrigatorio
48      const { id } = event.pathParameters;
49      Client.findById(id).then((client) => {
50        client.setData(body);
51
52        client.update().then(() => {
53          callback(null, apiResponse(201, {
54            body: {
55              client: client.toJson(),
56            },
57          }));
58        }) catch(error) => {
59
serverless.yml
178  --# Action:
179  --# provider: apigateway:GET
180  --# Resource:
181  --# arn:aws:apigateway:${self:provider.environment.region}::/apikeys/*
182  --# arn:aws:apigateway:${self:provider.environment.region}::/apikeys
183
184
185  package:
186    --# individually: true
187    --exclude:
188      --# .serverless/**
189      --# .git/**
190      --# node_modules/**
191      --# examples/**
192      --# awsEvents/**
193
194  stepFunctions:
195    --# stateMachines:
196    --# NameCheckerStepFunction:
197    --#   ${file(stepFunctions/nameCheckerStateMachine.yml)}
198
199  functions:
200    --# DRIVER-PROCESSING
201    --# SaveDriverBefore:
202    --#   handler: src/drivers.saveDriverBefore
203    --#   role: ${self:provider.environment.crawlingRole}
204
205    --# DriverNameCheckUp:
206    --#   handler: src/scrapers/nameChecker/crawler.driverNameCheckUp
207    --#   role: ${self:provider.environment.crawlingRole}
208    --#   timeout: 600
209
210    --# NameCheckerRJJF:
211    --#   handler: src/scrapers/nameChecker/crawler.crawl
212    --#   role: ${self:provider.environment.crawlingRole}
213    --#   memorySize: 1536
214    --#   timeout: 300
215
216    --# NameCheckerDFTJ:
217    --#   handler: src/scrapers/nameChecker/crawler.crawl
218    --#   role: ${self:provider.environment.crawlingRole}
219    --#   memorySize: 1536
220    --#   timeout: 300
221
222    --# PrepareNewNameCheckerAttempt:
223    --#   handler: src/scrapers/nameChecker/crawler.prepareNewNameCheckerAttempt
224    --#   role: ${self:provider.environment.crawlingRole}
225
226    --# SaveDriver:
227    --#   handler: src/drivers.saveDriver
228    --#   role: ${self:provider.environment.crawlingRole}
229
230    --# PriorityCrawlerConsolidator:
231    --#   handler: src/crawlerManagement.priorityCrawlerConsolidator
232    --#   role: ${self:provider.environment.crawlingRole}
233
234    --# PriorityOnlyFinalState:
235    --#   handler: src/crawlerManagement.priorityOnlyFinalState
236
```

Line 205, Column 21 | master UTF-8 Spaces: 2 YAML

Obrigado

Contatos:

- Ariel Schvartz
- E-mail: ariel@ariels.com.br
- LinkedIn: [linkedin.com/in/arielschvartz](https://www.linkedin.com/in/arielschvartz)
- Telefone: +55 21 99631-8859

Estamos sempre contratando! =)

Só entrar em contato!