Microservices em Python: Resiliência e Escalabilidade

Guilherme Vierno

About me

- Físico por formação
- Pythonista desde 2010
- Mineiro agora morando no Rio
- Desenvolvedor na globo.com
  - Plataformas de publicação

Por que falar de microservices?

Interesse ao longo do tempo

Fonte: google trends

Por que falar de microservices?

- Tendência do mercado
- Implementado nos líderes de tecnologia do mundo (*Amazon, eBay, Netflix, Apple…*)
- Resolve bem problemas de desenvolvimento em larga escala

Intro

Do Monolito aos Serviços

globo.com **(Monolito)**

pré micro-serviços

globo.com **(Monolito)**

pré micro-serviços

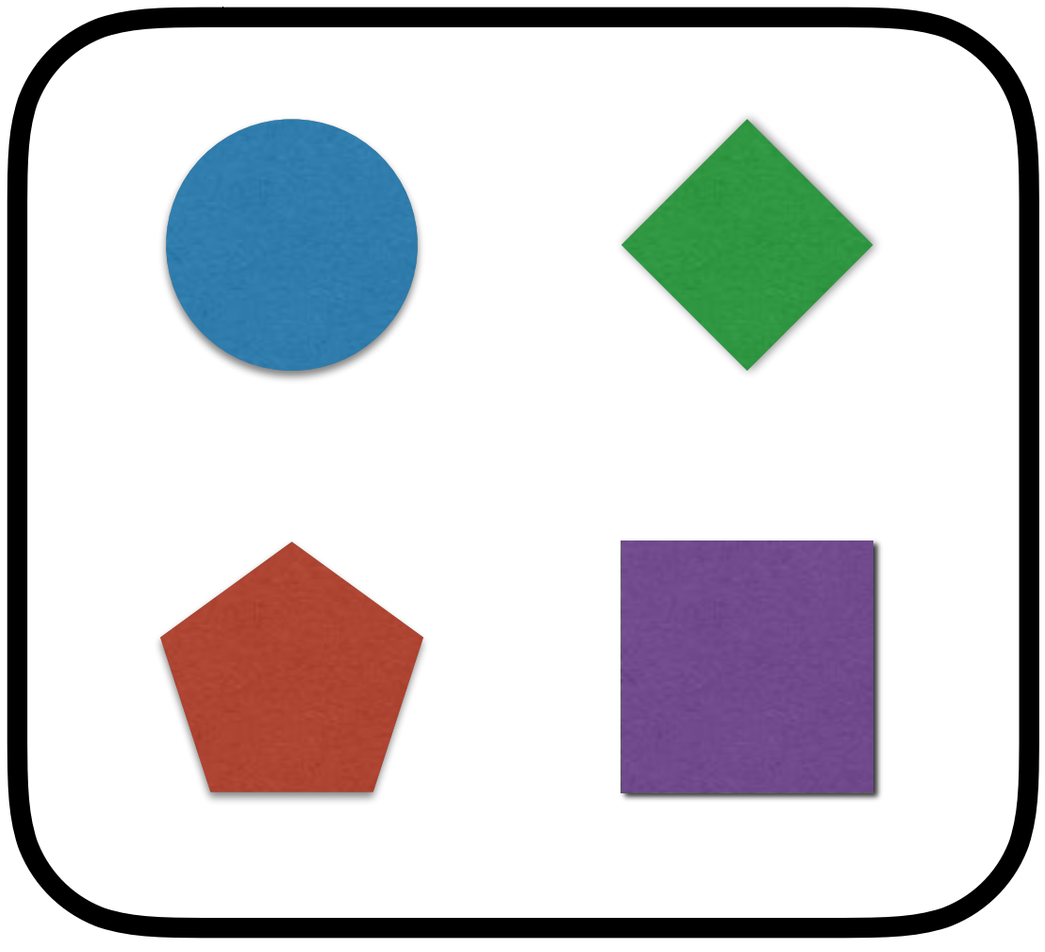
- Um **grande** projeto em Django
- ∞ linhas de código e milhares de testes
- Banco MySQL superinflado
- Vários times desenvolvendo o mesmo codebase

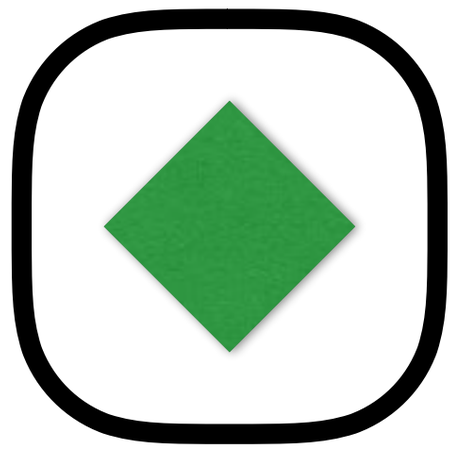
Monolito

Principais problemas

1. **Baixa velocidade de entregas**
    - “Time de deploy” e “dia do deploy”
    - Refatoração cada vez mais cara
2. **Escalabilidade**
    - Ruim de medir uso de recursos por módulo
    - Dificuldade em fazer cache eficiente
3. **Resiliência**
    - Algumas subidas causando instabilidade geral

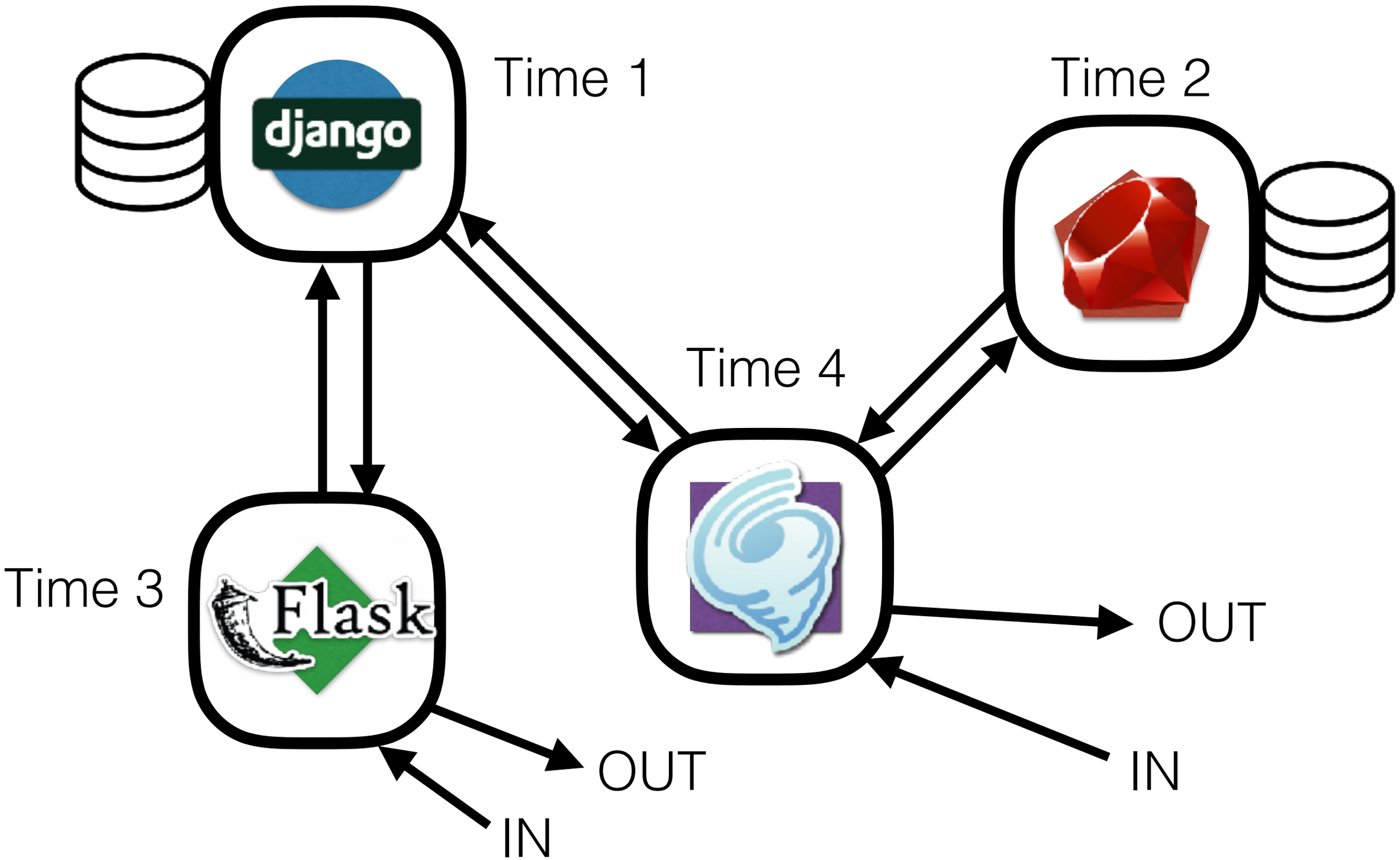
Separação em serviços





Migrando partes da aplicação para serviços isolados

Micro-serviços



Micro-serviços

Características comuns

- **Velocidade de desenvolvimento**
  - Pequenos em tamanho e limitados por contexto
  - Desenvolvidos autonomamente
- **Escalabilidade**
  - Deploys individuais
- **Resiliência**
  - Isolamento
  - Automatização de processos (build, release, monitoração…)

Python & Microserviços & Milhares de req/s & Alta disponibilidade

Python & Microserviços & Escalabilidade & Resiliência

1.Sistema eficiente de cache

g1.globo.com/.*

/noticias

Tornado

redis

mongoDB

/votar

mongoDB

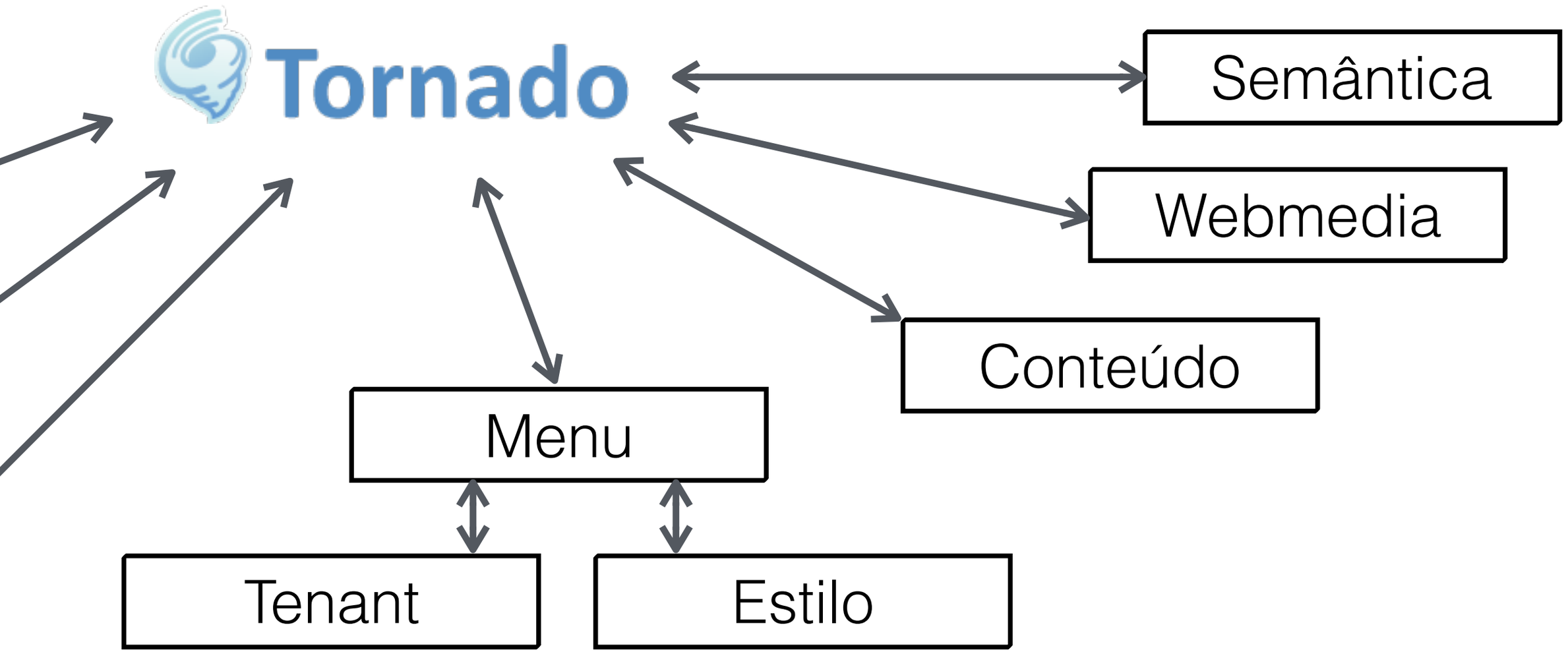
/busca

Tornado

elasticsearch

Renderizando uma matéria

`GET /noticia/tiroteio-deixa-um-morto-e-feridos.ghtml`

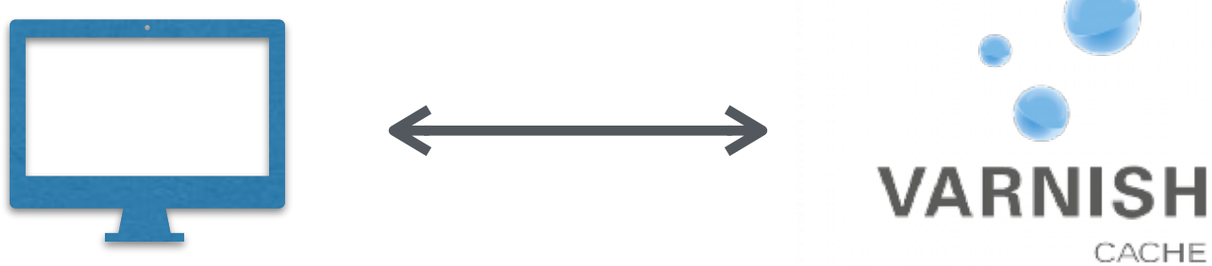


Cache distribuído

Cache distribuído

Cenário 1

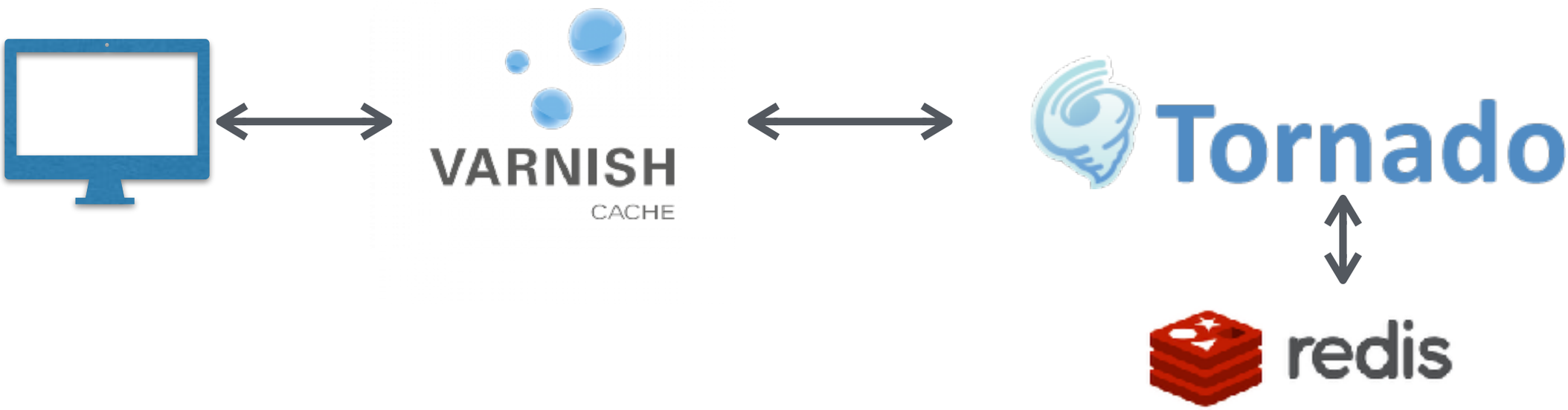
A página já foi requisitada neste servidor nos últimos 10 segundo



Cache distribuído

Cenário 2

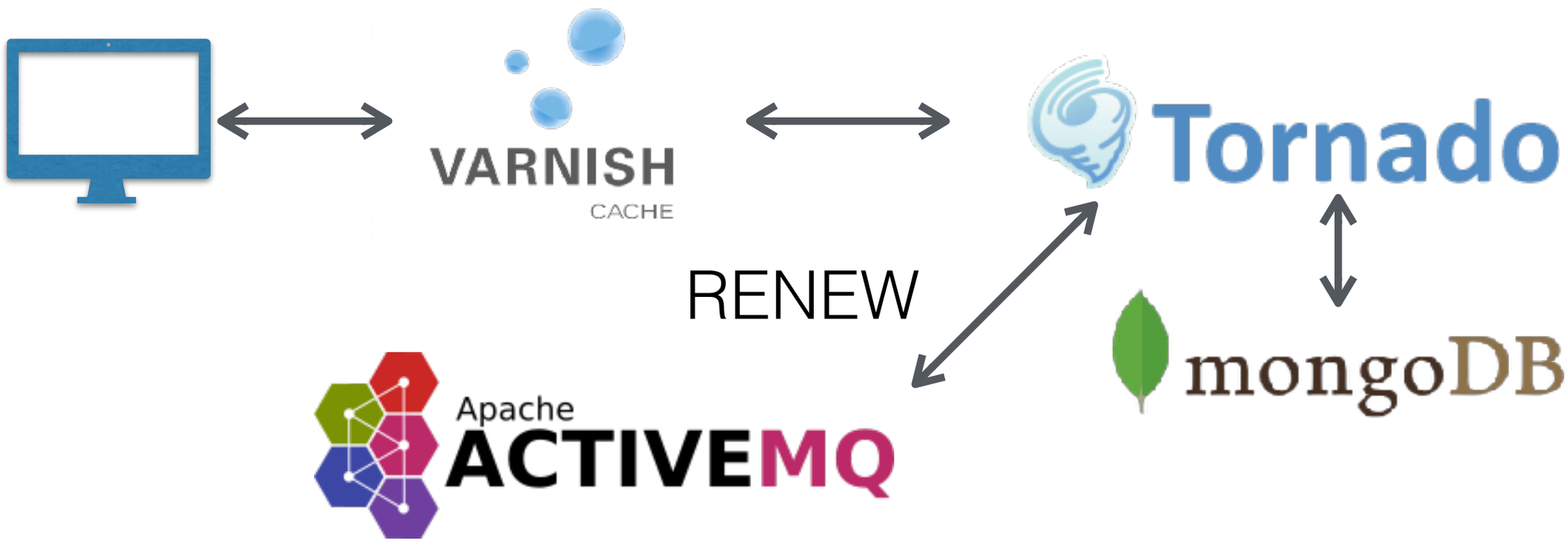
Cache do Varnish expirou mas a matéria foi acessada nos últimos 5 minutos e está fresca no redis



Cache distribuído

Cenário 3

A página não foi acessada nos últimos 5 minutos
ou não está no cache recente do redis

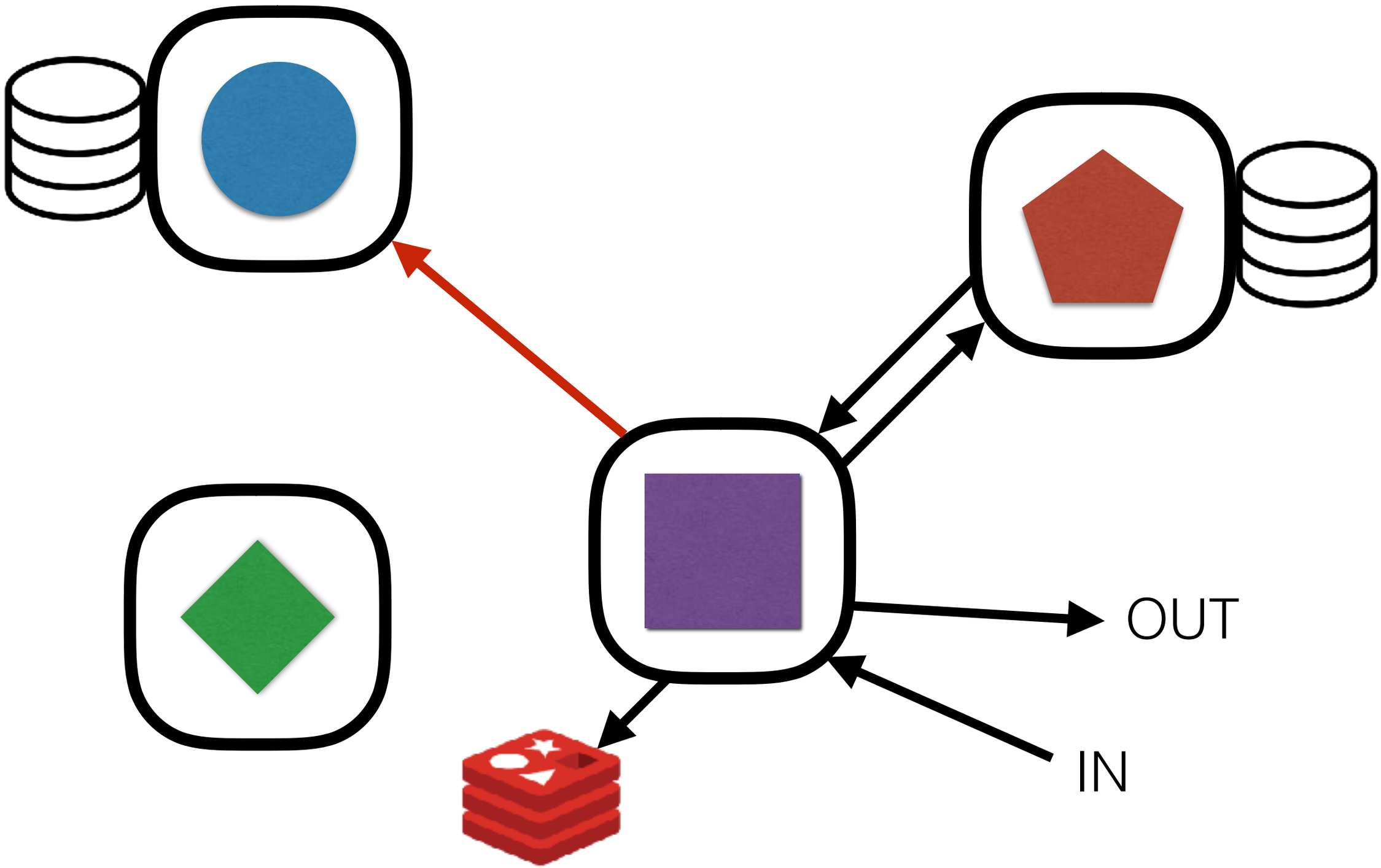


Cache distribuído

serviço	Tempo de cache	% das requisições servidas	Expiração
Varnish	10s	>80%	Passiva
Redis	5min	~15%	Ativa / LRU
MongoDB	5min - ∞	< 5%	Ativa

2.Expectativa de falhas

Circuit Breaking



Expectativa de falhas

- Cache + Stale (Fail-safe)
- Retry
- Backoff

3.Comunicação Leve e assíncrona

Bibliotecas assíncronas

- https://github.com/aio-libs/
    - aioredis
    - aiomysql
    - aiopg
- toredis
- tornado-alf

4.Monitoração

Healthcheck detalhado


```
{
   "status": "failure",
   "results": [
      {
         "output": "Connection Timeout",
         "checker": "redis_connection",
         "passed": false
      }
   ]
}
```

Ferramentas

ZABBIX

Cachet

New Relic®

SENTRY

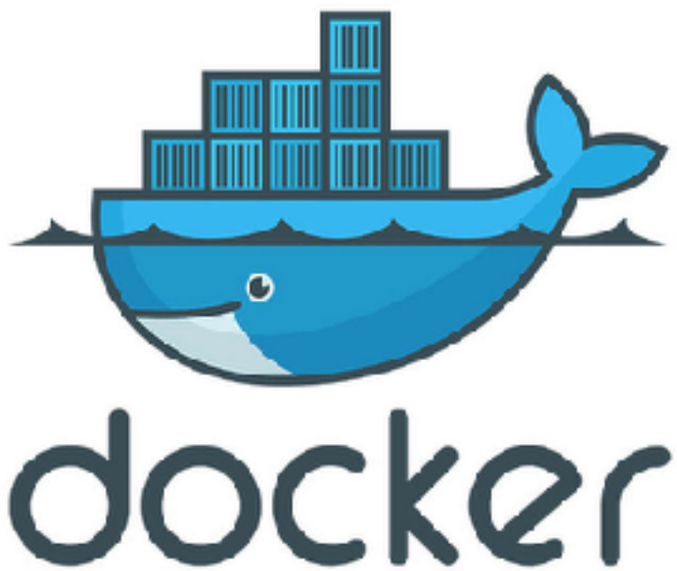
graylog

Grafana

5.Automação

Facilidade de subir Apps

- Isolamento das apps
- Automação de Infra
- FaaS & PaaS & IaaS
- Autoscaling
- CI / CD



Recap

1. Cache distribuído
2. Expectativa de falhas
3. Comunicação leve e assíncrona
4. Monitoração
5. Automação

E por que Python?

μservices em Python

Vantagens de usar Python

- Prototipagem e desenvolvimento rápidos
- Facilidade de instrumentar *continuous delivery / DevOps / automações em geral*
- Uso bem difundido de Microframeworks
- É bem fácil fazer comunicação leve entre apps

μservices em Python

Mais coisas legais

- Podemos combinar com outras linguagens
- Manutenção barata
- Uma boa hora de introduzir Python[3] na stack

μservices em Python

Frameworks

❤️ **Tornado:** Network I/O não bloqueante

- **Flask:** Zero ao deploy em pouco tempo
- **Django:** Tastypie, django rest framework, channels
- **Nameko:** Feito para microservices
- **aiohttp:** *asyncio* nativo com network

Considerações Finais

- *Monolith-first*
  - O momento certo é quando sentir a necessidade
- Atenção aos requisitos: Provisionamento de infra, monitoração, automação…
- A maior mudança é cultural

Isso é tudo pessoal!

We're hiring!
https://talentos.globo.com

Guilherme Vierno
vierno.com.br

Nem tudo são flores

- Complexidade de desenvolvimento/operacional
- Duplicação de dados
- Latência adicional
- Desuniformidade
- Refatoração cross-módulo