

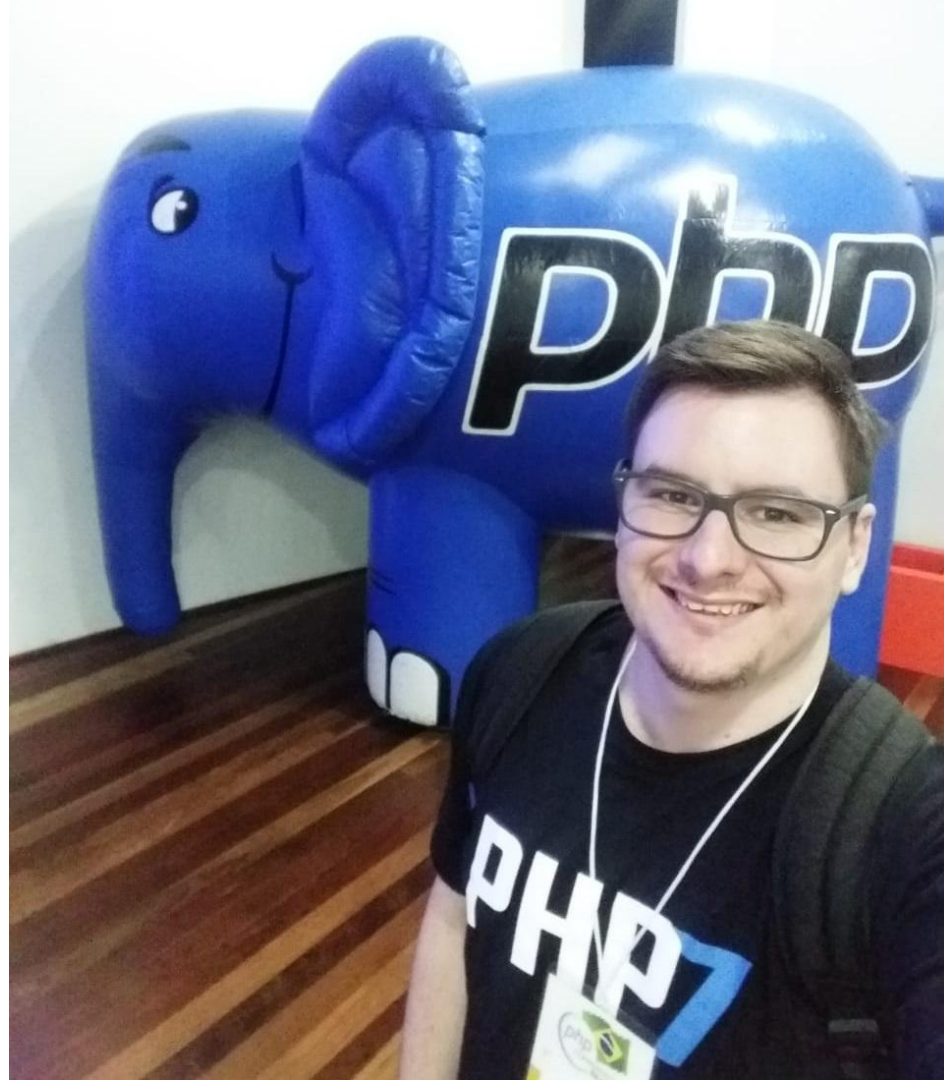


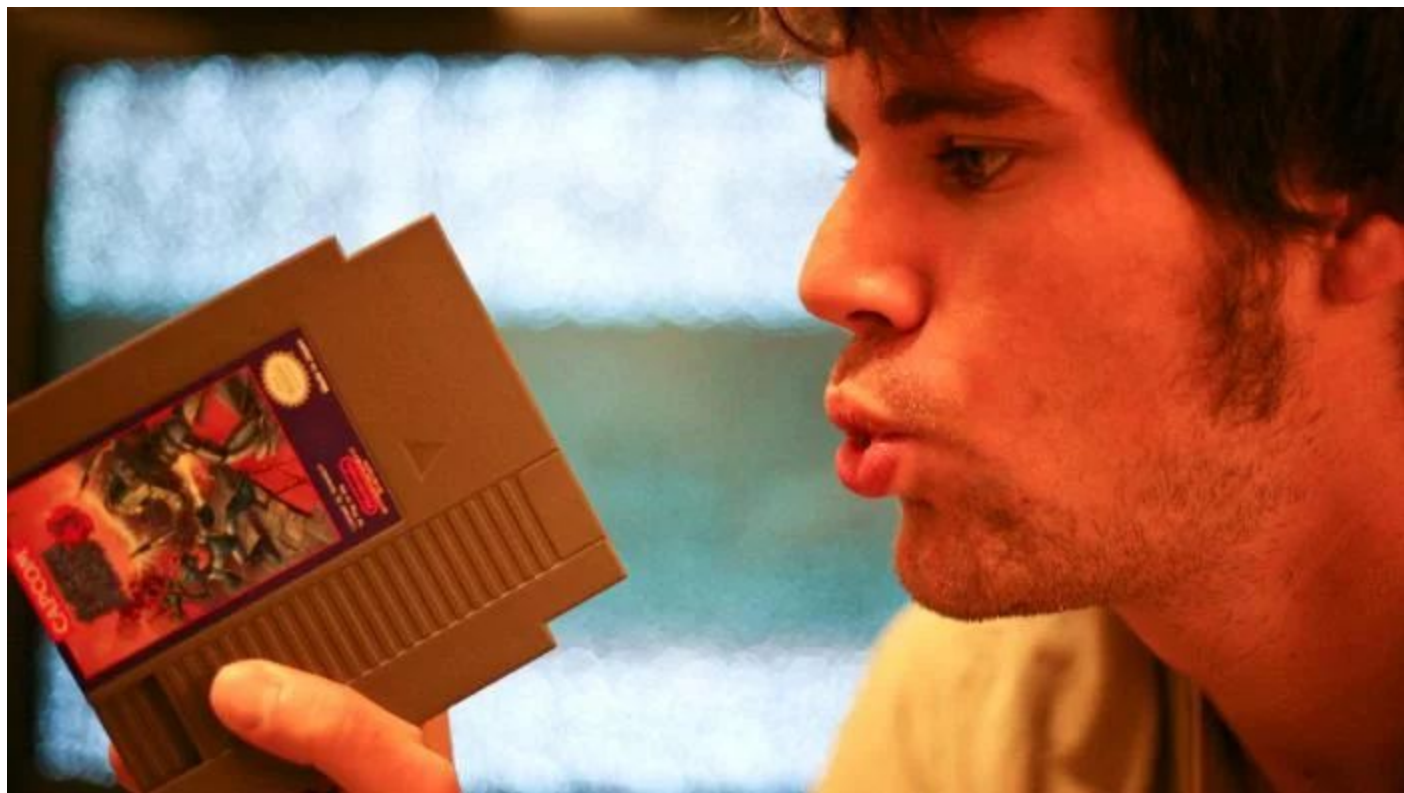
# **Monolíticos Distribuídos**

**Um case de falha que deu certo**

# PHPRS

- Sistemas de Informação - ULBRA
- PHP Developer  DATUM.TI.
- PHP Developer há 5 anos
- Coordenador da Comunidade PHPRS
- Escritor
- Criador de ElePHPants 
- Gamer nas horas vagas ;)





# Caso de Uso

- Aplicação monolítica legada
- Escrita em PHP 5.4 em 2012
- Totalmente estruturada e sem OO
- Dividida em 2 partes Site e CMS
- + 3 mil usuários cadastrados
- + 350 palestrantes
- + de 400 Oficinas, workshops e outras atividades
- 6 anos de congresso

# Cenário Atual

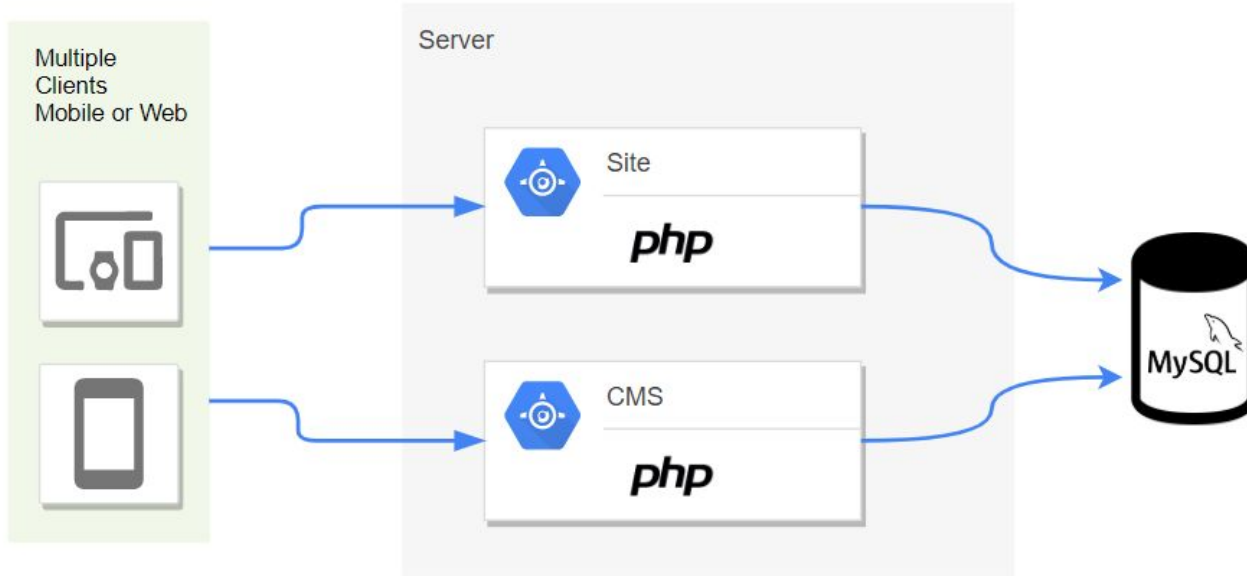
## Site

- Gerenciamento de usuários
- Inscrições e Pagamentos
- Certificados

## CMS

- Gerenciamento do conteúdo do Site
- Gerenciamento do Congresso
- Gerenciamento de Inscrições e Pagamentos

# Cenário Atual



# Porque Microserviços?

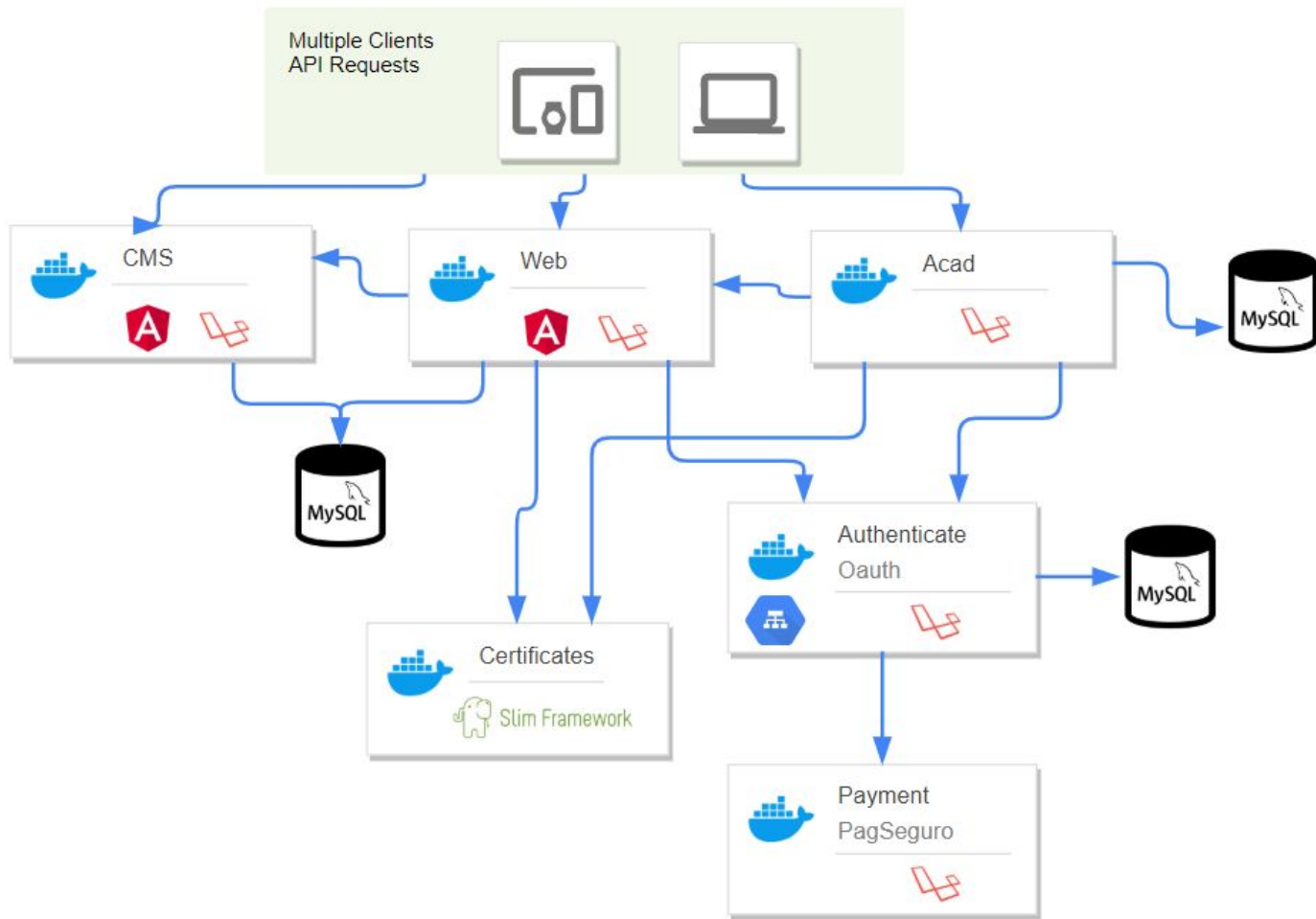
- Separar as responsabilidades em serviços independentes
- Estrutura descentralizada
- Facilidade na manutenção
- Facilidade e Agilidade para o desenvolvimento de novas features
- Facilidade para subir ambiente usando Docker
- Estava no **HYPE**

# **Cenário Atual**

**+**

# **Novas Features**





# Beleza! e Agora?

# #timetocode





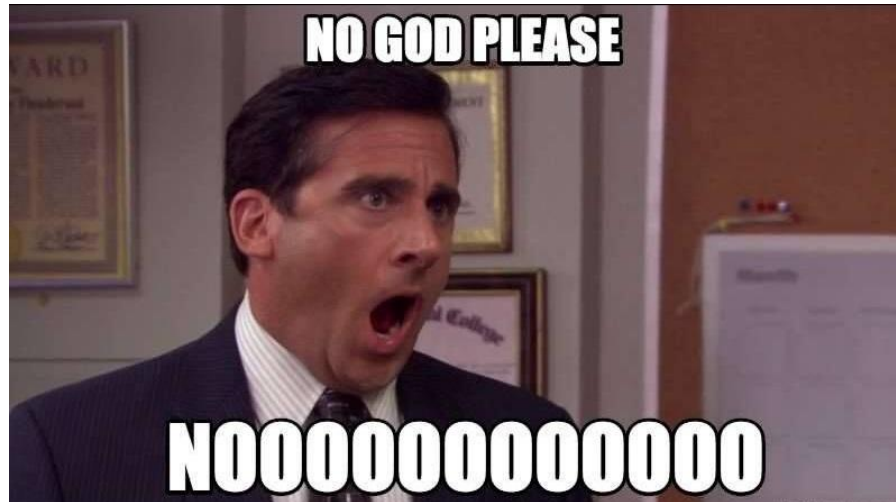
4 meses depois





Projeto Entregue

**Sim! e com ele  
diversos problemas!**



# Logs

Os logs estavam todos em nível de aplicação.  
Ou seja cada serviço tratava os seus logs a "sua maneira"

---

Uma arquitetura de microsserviços distribuídos é muito difícil de se monitorar.

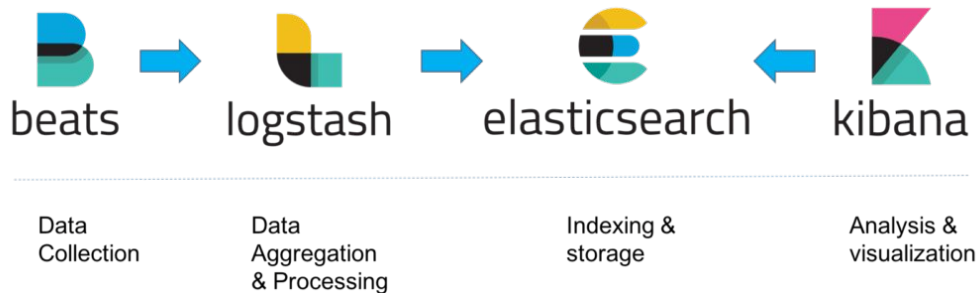
Se acontece algum problema ou algum erro em algum serviço:

- Como vamos saber o que aconteceu?
- Como vamos saber aonde aconteceu?
- Em qual fluxo o usuário estava?

Neste cenários os logs de arquivos vão ser o seu pior inimigo.

# Qual seria a solução?

## Monolog - Logging for PHP





# Gargalos

Em alguns momentos ocorreu lentidão ou falhas, principalmente ao realizar pagamentos. Por causa do grande número de requisições ao mesmo tempo.

---

Todas requisições estavam sendo processadas de forma Síncrona e a melhor forma de resolver isso seria trabalhar a comunicação de forma Assíncrona para facilitar a comunicação entre serviços.

# Qual seria a solução?

Implementação de algum padrão como Publish Subscribe.



# Melhorias

# API Gateway

Serve como a porta de entrada para os seus microsserviços resolvendo problemas comuns como: Autenticação, Limite de uso e CORS (Cross-origin).

## Sem Gateway

Cada serviço implementa sua lógica de autenticação, Log e Cache por exemplo, que acabam gerando inconsistência na utilização de API's.

## Com Gateway

Autenticação, Log e Cache podem ser resolvidos por exemplo, em uma camada antes de chegar em cada serviço simplificando e centralizando a utilização das API's.

# API Gateway

## Desvantagens

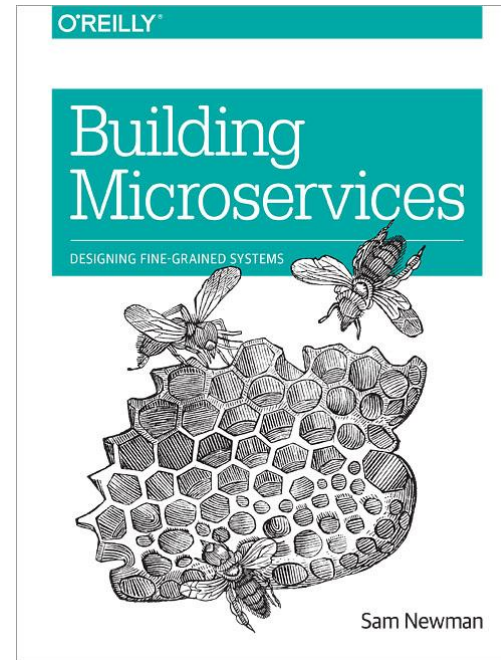
- Mais um serviço a ser desenvolvido e mantido.
- Pode se tornar um gargalo de desenvolvimento.

## Vantagens

- Único ponto de entrada.
- Pode mascarar falhas ou erros, retornando cache de dados.
- Fornece maior segurança.

**Aprendizado  
imensurável  
para a equipe!**

# Recomendação



# Dúvidas





[twitter.com/IgorSantoos17](https://twitter.com/IgorSantoos17)



[linkedin.com/in/igorsantoos](https://linkedin.com/in/igorsantoos)



[github.com/IgorSantos17](https://github.com/IgorSantos17)



[medium.com/@igorsantos17](https://medium.com/@igorsantos17)



[speakerdeck.com/igorsantos](https://speakerdeck.com/igorsantos)





#junteseamanada