



Foco no app, viva o  
serverless!

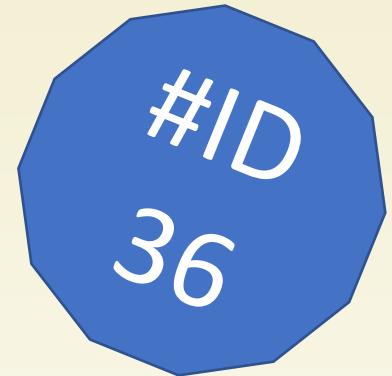
Mario Guedes - #36

Da sala de reunião à vida real

Embarcadero Conference 2019

# Mario?

- **Codificador** Delphi desde 1999 – Vinte anos e contando!
  - Obrigado Rodrigo!
- Fui **Instrutor** Delphi na TDS de 2011 à 2013
  - Obrigado Leandro!
- **Palestrante** na EC, e outros eventos, desde 2011 – Nono ano!
  - Obrigado Andreano!
- Fui **Colunista** na Active Delphi de 2012 à 2014
  - Obrigado Kelver!
- **Embarcadero MVP** desde 2016 – Quarto ano!
  - Obrigado Marcão!

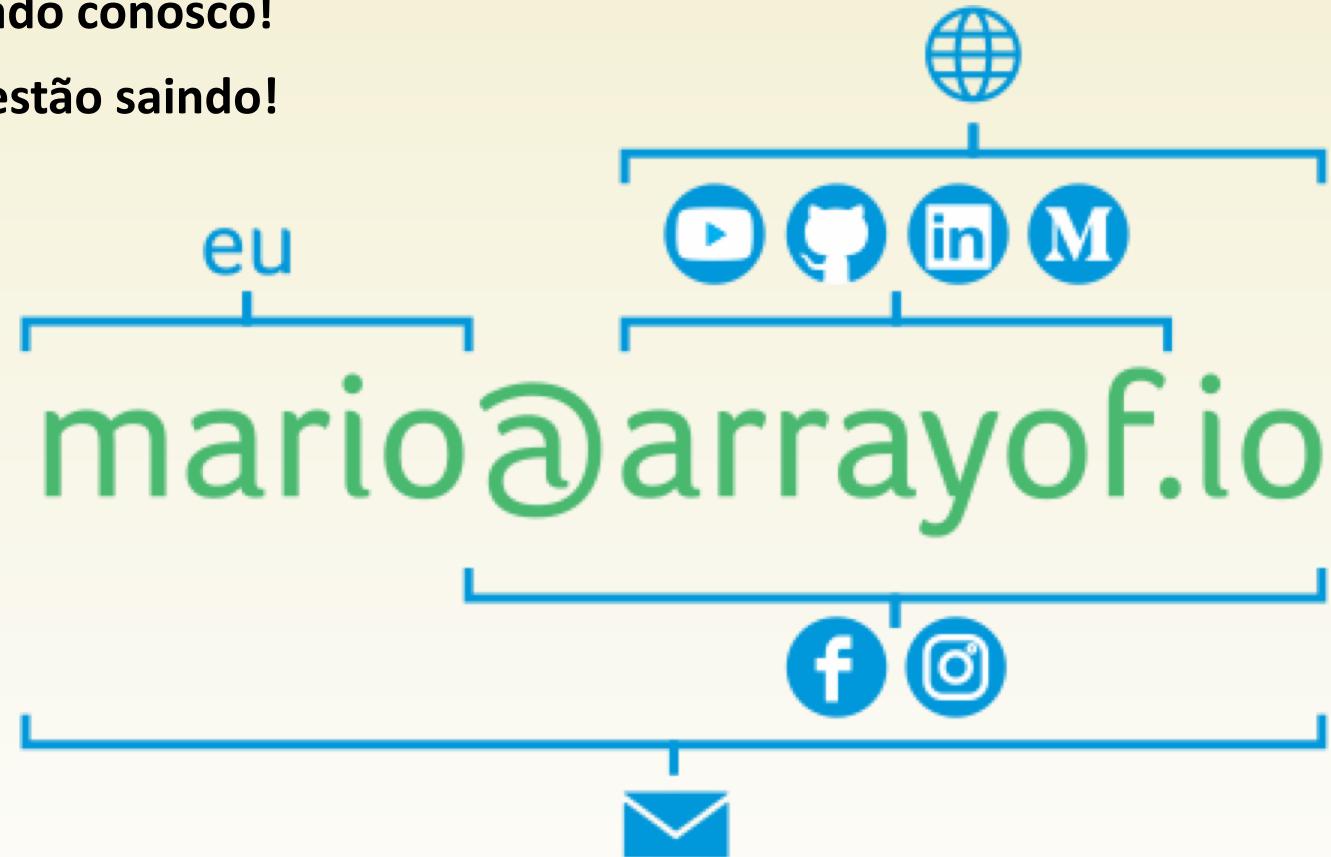


Mario Guedes – #ID: 36

embarcadero®

# Siga a ArrayOf.io

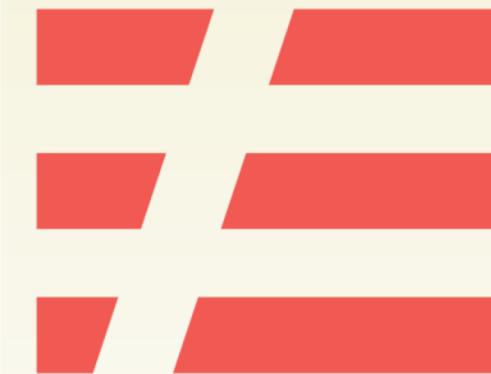
- Modernize o seu legado conosco!
- Os primeiros cursos estão saindo!



Mario Guedes – #ID: 36

embarcadero®

Hoje falaremos de:



# serverless



Mario Guedes – #ID: 36

embarcadero®

# Pesquisa rápida

- Quantos aqui são Administradores de Rede?
- Quantos aqui são DBA?
- Quantos aqui são DevOps, SysAdmin e por aí vai?
- Quantos aqui manjam de servidores e Sistema Operacional profundamente?
  - Windows?
  - Linux?
- **Agora: Quantos aqui são desenvolvedores?**



Mario Guedes – #ID: 36

# BaaS + FaaS = *Serverless*

- **IaaS:** *Infra estrutura como serviço*  
*Quando contratamos servidores virtuais estamos usando o IaaS.*  
Aqui ainda não temos o *serverless* em ação.  
e.g. EC2 da AWS
- **BaaS:** *Backend como serviço*  
*Mas quando contratamos um banco de dados ou um serviço de autenticação por exemplo, estamos lançando mão do BaaS. Mas só é verdade se este for acessível diretamente ao front-end.*  
e.g. Firebase do Google
- **FaaS:** *Função como serviço*  
*Agora, se um serviço nos permite subir um código e este código ser executado sob demanda – aí estamos usando o FaaS.*  
e.g. AWS Lambda da Amazon, Cloud Functions da Google ou Azure Functions da Microsoft



- **Serverless:** *Sem servidor*
- Você escolhe um provedor de serviços (Amazon, Google, Microsoft entre outras menores)
- Explora os diversos serviços nas seguintes áreas:
  - Execução de código
  - Armazenamento de dados e arquivos
  - Serviço REST
- Tudo isto sem levantar nenhuma máquina, seja física ou virtual
- Você paga pelo recurso que usar e quando usar
- Hoje iremos explorar alguns serviços da Amazon.
  - *E não é propaganda OK? É aquela que uso e conheço.*



# Qual a dor que o *serverless* resolve?

- Imagine um solução baseada em aplicativo **web, mobile ou IoT**
- Você terá que desenvolver uma aplicação que consumirá, necessariamente, um **backend**
- Portanto você terá que desenvolver a aplicação de backend se preocupando com a **escalabilidade** da coisa toda
- Logo você terá que utilizar um **banco de dados** para armazenar os dados
- E eventualmente uma máquina para **armazenamento** de arquivos – fotos, XMLs e etc
- Como é uma solução *cloud* você terá que contratar um **datacenter** e terá que comprar/alocar equipamentos
- Por **segurança** você criará dois ambientes



## Problemas levantados:

- Custos de contratação, aquisição, configuração e manutenção de servidores
- Tempo de desenvolvimento dos serviços de backend
- Custos de propriedade dos servidores pois nos primeiros meses ou anos eles estarão subtilizados
- Mas você tem que deixa-las ligadas *full time* afinal nunca se sabe quando um usuário se logará
- *E você não tem este dinheiro nem o conhecimento necessário para lidar com isso tudo porque você é programador e só quer colocar a sua solução no ar*
- *Seria tão bom se você não precisasse se preocupar com servidores ...*



# Foco no app ...

- Você tem que focar no que o seu aplicativo se propõe a fazer de melhor
- E *terceirizar* aspectos que fogem ao propósito primordial da sua solução
- ***Use e abuse de todo o seu conhecimento em Delphi para desenvolver a melhor solução do seu segmento de mercado***
- Com isso você ganha velocidade de desenvolvimento e, consequentemente, lançamento do seu produto no mercado
- Aquilo que você “*terceirizar*” será consumido via API REST



# *... viva o serverless!*

- Vamos conhecer 5 serviços da AWS:
  - AWS Lambda
  - AWS API Gateway
  - AWS DynamoDB
  - AWS S3
  - AWS Cognito
- Porém o **Google Platform Cloud** e a **Microsoft Azure** oferecem serviços com os mesmos propósitos
  - Escolha a que mais lhe agradar
  - *Preço, camada free, familiaridade, confiabilidade são bons critérios de escolha*
  - Cuidado com o *vendor-lock*: a migração entre plataformas ainda é um processo doloroso – mas há iniciativas a respeito



# Recursos Compartilhados X Dedicados

- Já sabemos que a AWS oferece máquinas virtuais e instâncias de banco de dados (SQL Server por exemplo). Qual é a diferença?
- A diferença é que a AWS EC2 ou a AWS RDS são **recursos dedicados**:
  - Você paga pelo fato deles estarem ativos, não importa se está sendo utilizado ou não
- Já serviços como AWS Lambda ou AWS DynamoDB são **recursos compartilhados**:
  - Por isso você paga apenas por aquilo que foi utilizado
  - A camada free é generosa – você poderá ficar um bom tempo sem gastar nada ou gastar muito pouco – mesmo com a solução em produção



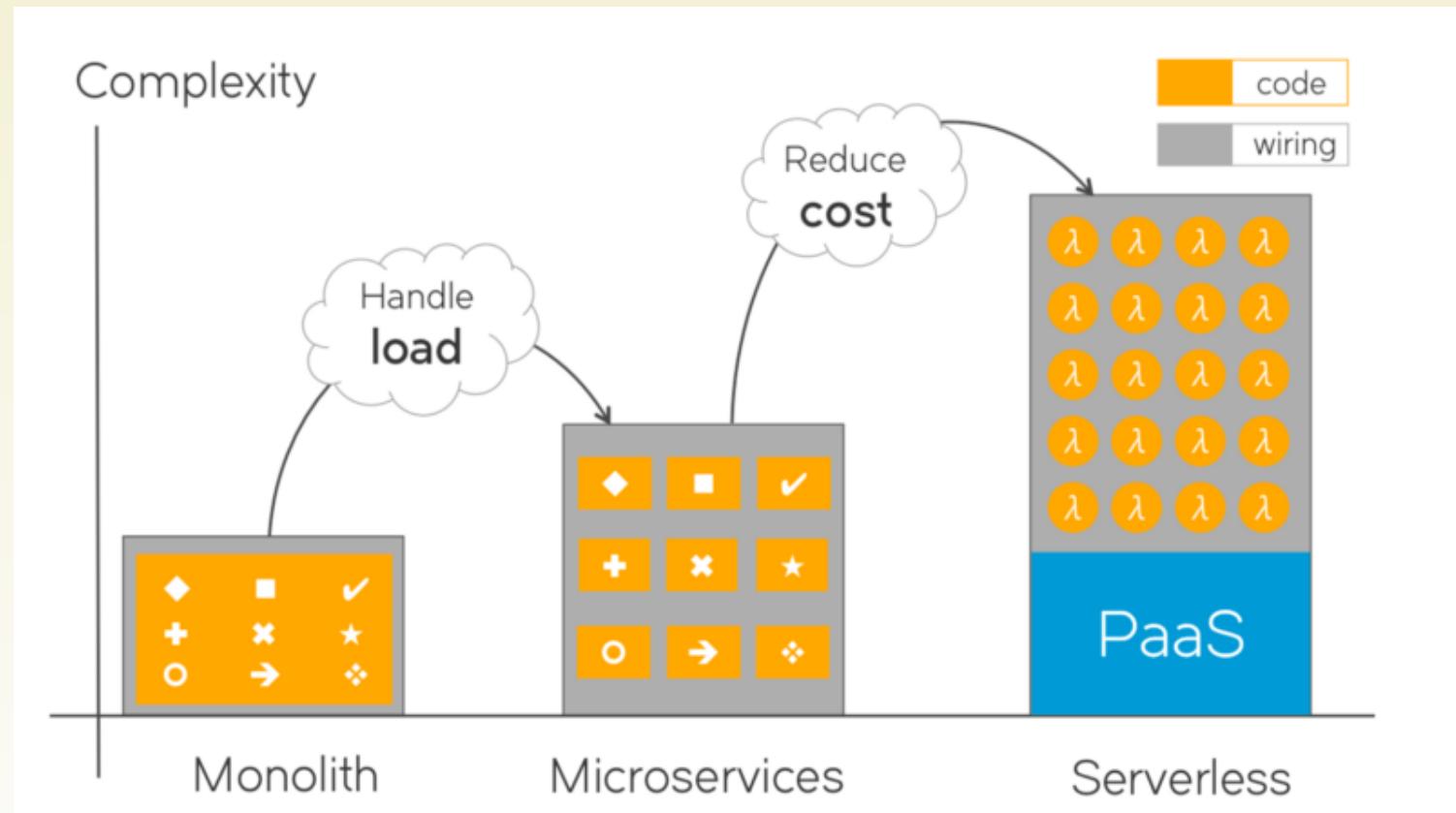
# AWS Lambda

- Aqui você codificará as suas regras de negócio
- Oferece várias linguagens mas sugere-se Python ou NodeJS pois são as que tem menores *cold start*.
  - *A pior é Java, pra variar*
- Evite um grande acoplamento ao ambiente da AWS para facilitar uma eventual migração.
- Essas funções são executadas em *contêineres efêmeros*:
  - *Contêiner com pouco tempo de vida que sobe, executa o código e é descartada logo em seguida*
  - *Por isso que o cold start é importante: impacta no tempo global da execução*



# Monolítico X Microserviços X Nanoserviços

- As funções também são conhecidos por *nanoserviços*
- Você codifica somente e tão somente o necessário para atender a sua regra de negócio:
  - Não precisa se preocupar com HTTP, Thread, Instância disso ou daquilo ...
  - Direto ao ponto!



# AWS API Gateway

- Equivalente ao DataSnap, manja?
- Você configura os *endpoints* da sua API e direciona o conteúdo para algum outro serviço
- *Pode vir a quantidade que for de requisições que será atendida!*
  - Mas se não vier nenhuma tudo bem.
- Você define o direcionamento que será dado à requisição HTTP:
  - Uma função Lambda?
  - Um endpoint em uma máquina EC2?
  - Enfileirar no AWS-SQS? (Mensageria da AWS)
  - Algun outro serviço?
- Daí responde ao chamador.



# AWS DynamoDB

- Equivale, mais ou menos, à uma mistura de Redis com MongoDB
- É um banco de dados noSQL orientado à chave e valor
- A monetização é baseada – principalmente - em unidade de leitura e escrita: **RCU** e **WCU**

## Cenário de exemplo:

Uma aplicação recebe **3 milhões de leituras** por mês e **500 mil de escrita**.

## O cálculo seria:

$3.000.000 / 30 \text{ dias} / 24 \text{ horas} / 60 \text{ minutos} / 60 \text{ segundos} / 2 \text{ por unidade} \approx 1 \text{ unidade de escrita por segundo}$

$2.000.000 / 30 / 24 / 60 / 60 \approx 1 \text{ unidade de escrita por segundo}$

## Isso perfaz o valor de:

1 unidade leitura x US\$ 0,09 == US\$ 0,09

1 unidade de escrita x US\$ 0,47 == US\$ 0,47

**Soma: US\$ 0,56**



# AWS S3

- Equivale à um servidor FTP da vida.
- Você armazena, essencialmente, arquivos.
- Bacana que você pode hospedar site estático ou até mesmo um SPA ou PWA desenvolvido em Angular ou React por exemplo.
- É como a zueira no BR: *Sem limites*



# AWS Cognito

- Podemos utilizar o AWS Cognito para gerir as autenticações e permissões de usuários.
- Associado ao AWS IAM podemos definir todas as regras de acesso.
- É ~~foda~~ difícil entender no começo mas depois fica entra na veia



# DEMOSTRAÇÃO



Mario Guedes – #ID: 36

embarcadero®

# Outros serviços

- A AWS, bem como os outros provedores, possuem diversos outros serviços que podem e devem ser explorados
- Basicamente falamos de uma **arquitetura orientada a eventos.**



# Case em desenvolvimento

- Software house que desenvolve sistema para lojas de móveis
  - *+20 anos de estrada*
- Sistema *on premise* ou seja – instalado na casa do cliente (servidor e tudo o mais)
- Clientes querendo ir para a Internet, aplicativo – enfim ...
- Concorrente babando ...
- **Solução:**
  - Criação de um ambiente na AWS: Lambda + API Gateway + Dynamo + S3
  - Criação de um aplicativo de força de vendas
  - Legado alimenta o Dynamo: produto x preço
  - Legado alimenta o S3: fotos dos produtos
  - Criação de um aplicativo Delphi que consome o ambiente AWS
  - Quando um aplicativo fecha uma venda, bate no Lambda que faz uma chamada ao DataSnap da loja



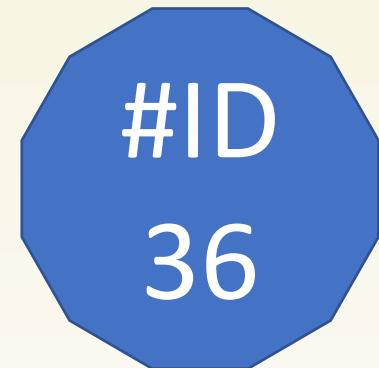
# Crie aplicativos com serviços, não servidores

- Talvez o *serveless* não resolva todo o seu problema mas talvez algum problema específico
- Infelizmente não há suporte à nossa linguagem preferida, Delphi – mas encaro isso como escrever stored procedures no banco ou uma macro no Excel: é da vida
- Sugiro Python como linguagem para o AWS Lambda pois é o que tem menor *cold start*, assim como o NodeJS.
- ***Cuidado com a documentação e gestão da coisa toda! Quem faz o que, quem chama o que, onde armazena a informação tal, permissões e por aí vai***
  - ***Codificar menos, gerenciar mais***
- Escolha bem a região: EUA = mais barato AND maior latência X BRASIL = mais caro AND menor latência
- Monitore constantemente o que esta acontecendo no seu ambiente para não ter surpresas na sua fatura.
  - *Não precisa ter medo, basta controlar.*



# Perguntas e considerações

- Se você é de **Porto Alegre** e região lembre-se que teremos o **TDC**, com a Trilha Delphi presente!
- **Sexta feira, 29/11 – Estaremos lá**
- <https://thedevelopersconference.com.br/tdc/2019/portoalegre/trilha-delphi>



Mario Guedes – #ID: 36

## Trilha Delphi

Um código, todos os dispositivos: Rapid Application Development para desktop, mobile e IoT.

Construa aplicações nativas para Android, IOS, MacOS, Linux e Windows com muita agilidade.

Nesta trila abordaremos o suporte a múltiplos banco de dados, BIGData, IOT e a utilização de API's REST.

Tudo isso com um único código.

## Coordenação



Juliomar Marchetti



Newton Oliveira



Samuel David

# Referências

- <https://medium.com/@dayvsonlima/você-sabe-o-que-é-arquitetura-serverless-1f6dd1184e5b>
- <https://www.slideshare.net/AmazonWebServices/serverless-for-developers>
- <https://medium.com/@oieduardorabelo/serverless-framework-as-10-melhores-pr%C3%A1ticas-recomendadas-e2ec59f92699>

*Referências consultadas em Outubro de 2019*



Mario Guedes – #ID: 36

embarcadero®



# Obrigado

@ [mario@arrayof.io](mailto:mario@arrayof.io)

f [/arrayof.io](https://arrayof.io)

t [/arrayof\\_io](https://arrayof_io)

i [/arrayof.io](https://arrayof.io)

Da sala de reunião à vida real

Embarcadero Conference 2019