



Como a plataforma de buscas do iFood funciona

Lucas Viecelli

PGConf.Brasil 2019

**1º A 3 DE AGOSTO
SÃO PAULO - SP**

**PALESTRAS | TUTORIAIS | LIGHTNING TALKS
+ 1 DIA DE WORKSHOPS**

PGCONF.COM.BR



Um pouco sobre o iFood

Infraestrutura

Alguns números:

+900

Instâncias no pico

+90

Instâncias de DB

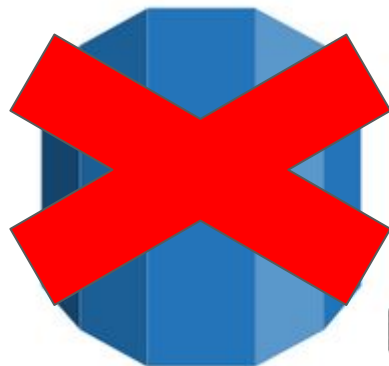
99%

Automatizado



Por que não o RDS?

- Sem acesso a todos os parâmetros do servidor, OS e database.
- Sem acesso remoto do host via ssh.
- Sem acesso ao super user.
- Não se pode instalar outras aplicações no host do banco, Ex: pgbouncer, pgpool.
- Falta de visibilidade



Amazon RDS



PostgreSQL



Então como fizemos nosso próprio RDS?

- Com configuration manager.
- Com infrastructure as code.
- Com shell script.
- Com scripts em python.



CHEF



HashiCorp

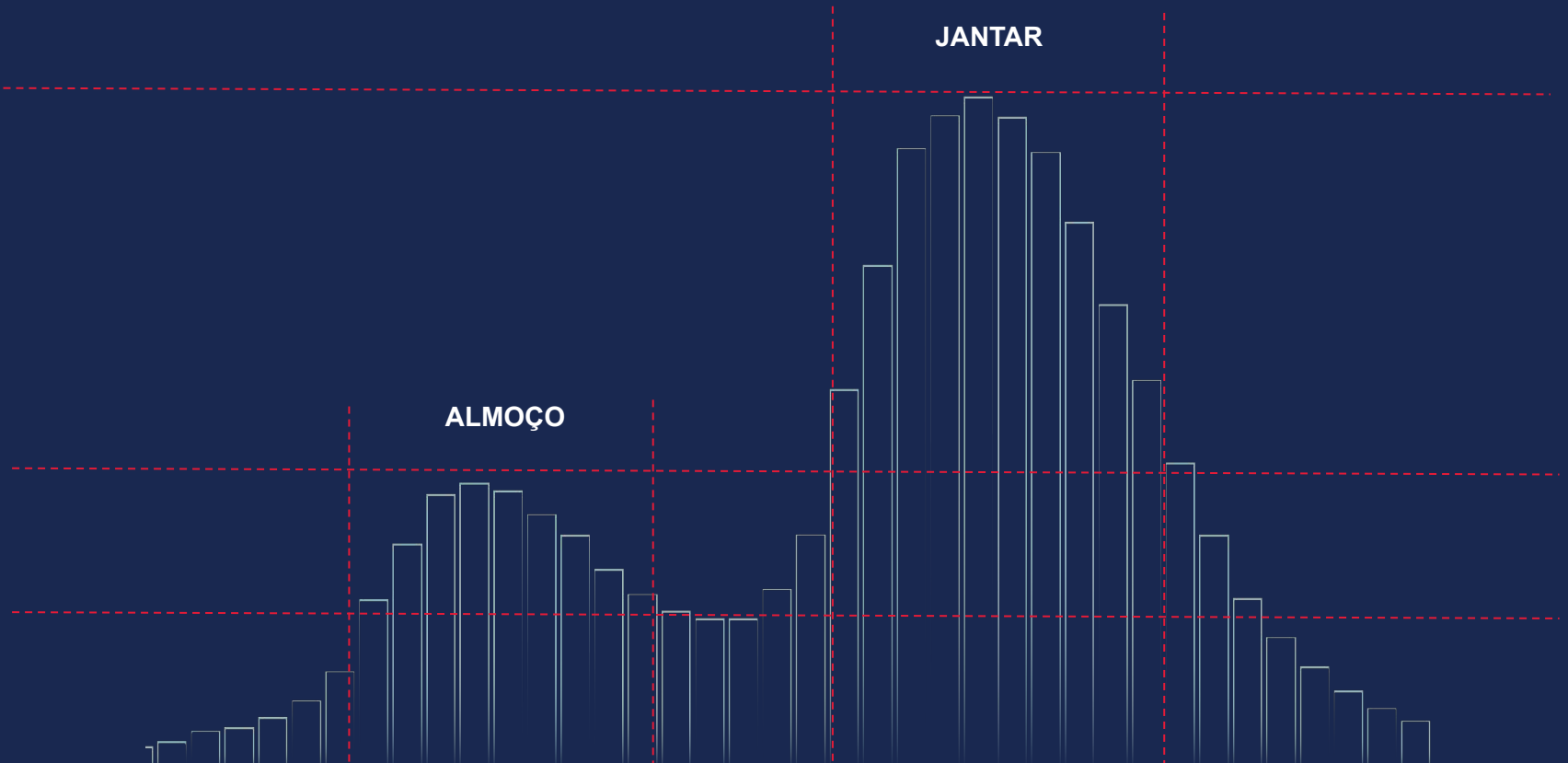
Terraform



python™

```
#!/bin/bash
```

Infraestrutura



Crescimento...



BRAZILIAN FOOD DELIVERY EXPANSION

GLOBAL MARKET NUMBERS

Current Revenue in Online Food Delivery
US\$82B in 2018 | +17.1% yoy

Future Revenue Expected
Growth rate (CAGR 2018-2023) of 10.7% → resulting in a market volume of **US\$137B** by 2023.

*according to Statista 2018

iFood orders per day in Brazil



ORDERS IN MARCH 2019

17.4 million

Present in **500 cities** in every state, in Brazil

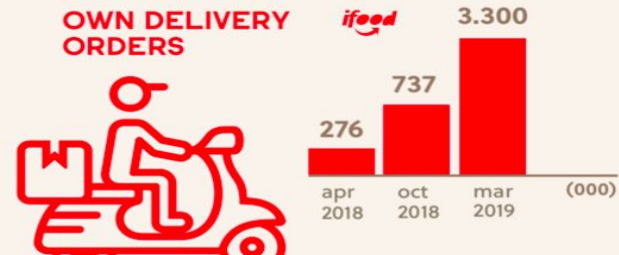
17x the nearest competitor * in terms of daily active users

*SimilarWeb data on Android App daily active users for January 2019

RESTAURANT GROWTH



OWN DELIVERY ORDERS



Plataforma de Busca

O problema: dados de vários serviços integrados na mesma tela

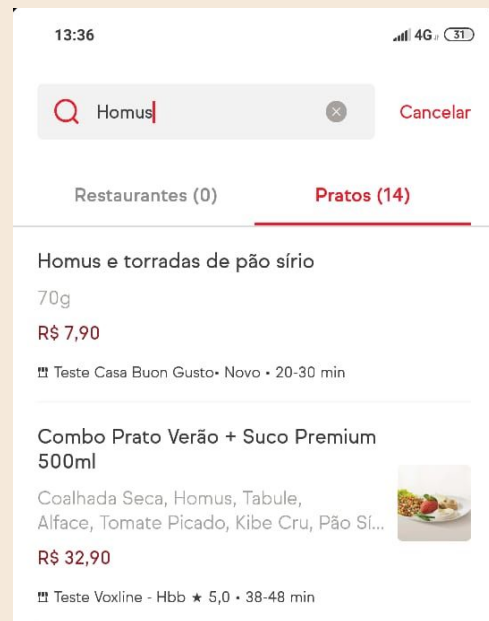
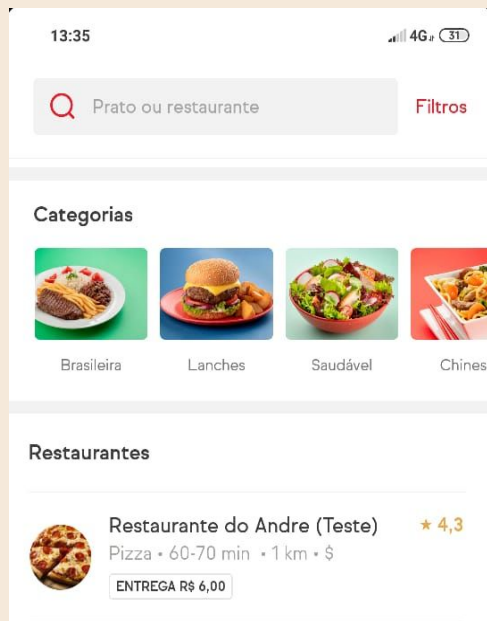
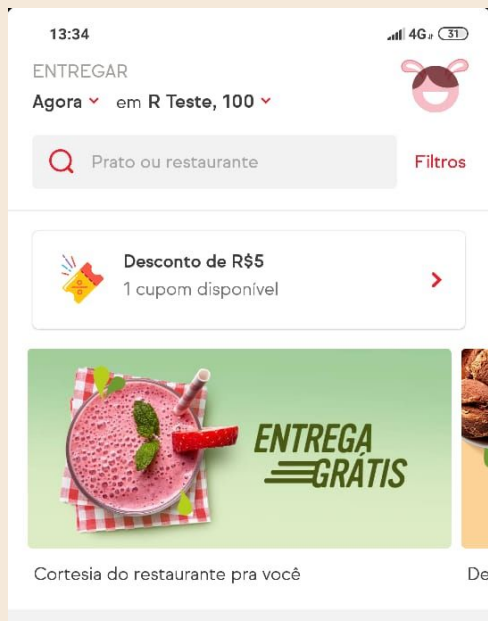
The image shows a mobile application interface for food delivery. The top status bar displays the time 20:25, signal strength, Wi-Fi, and battery level at 89%. The app header includes the word "ENTREGAR" and a location dropdown menu currently set to "Agora em R Teste, 100". A search bar contains the text "Prato ou restaurante" and a "Filtros" button. Below the search bar, a section titled "Restaurantes" lists three items:

- Modelo 9 Teste Pos2**: Pizza • 0-10 min • 0,3 km • \$\$, with a 4.0 star rating and a button for "ENTREGA VARIÁVEL".
- Teste Datamaxi**: Brasileira • 1-11 min • 0 km • \$, with no rating and a button for "ENTREGA R\$ 7,00".
- Restaurante lfood**: Alemã • 30-40 min • 0 km • \$, with no rating and a button for "ENTREGA GRÁTIS".

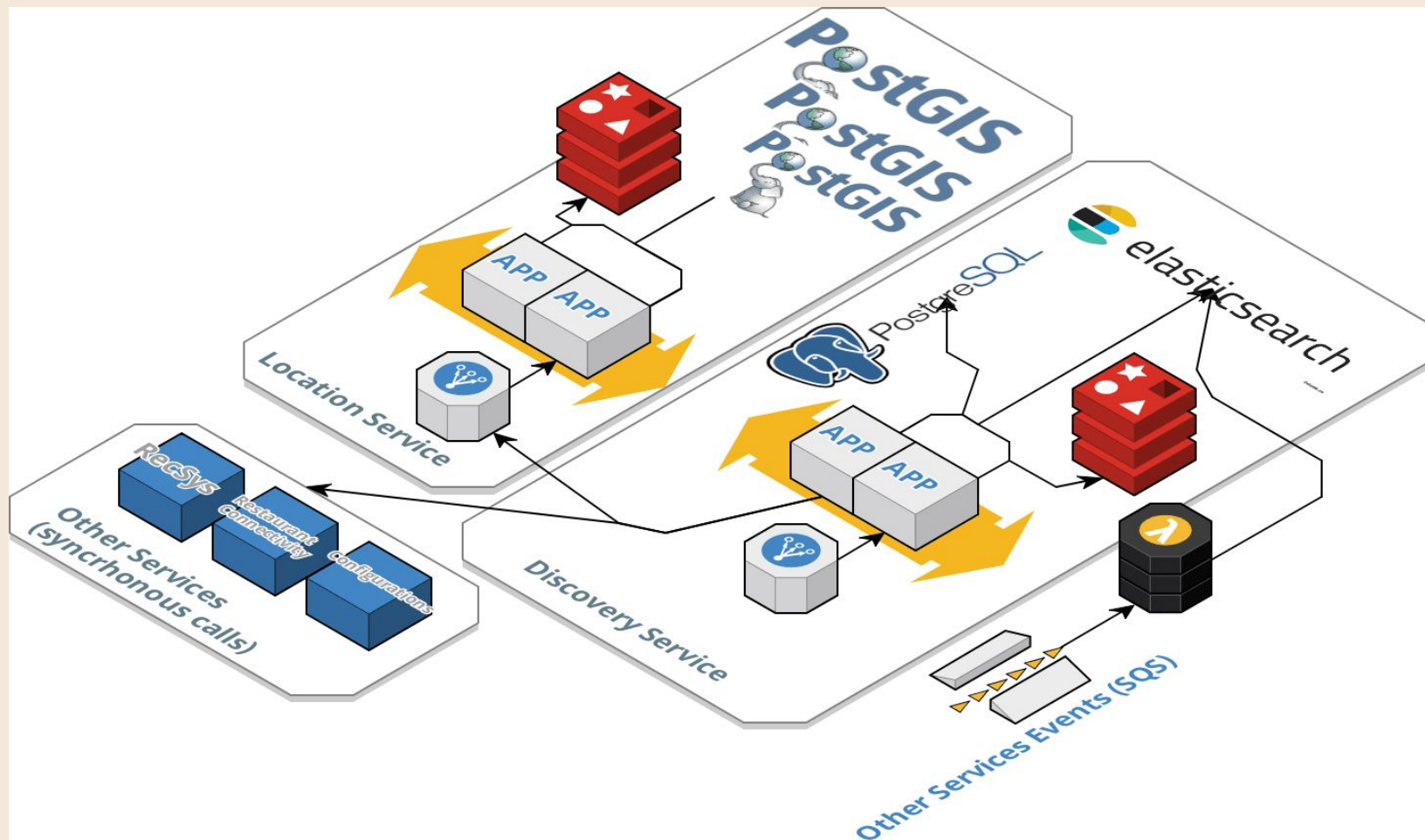
Annotations with arrows point to various elements:

- Lat/Lon**: Points to the location dropdown menu.
- Termo**: Points to the search bar.
- Restaurantes e pratos**: Points to the list of restaurant entries.
- Avaliação**: Points to the star rating of the first restaurant.
- Entrega**: Points to the delivery status button of the first restaurant.
- Tempo**: Points to the estimated delivery time of the first restaurant.
- Distância**: Points to the distance of the first restaurant.

Usamos essa estrutura em várias partes da app



Solução antiga

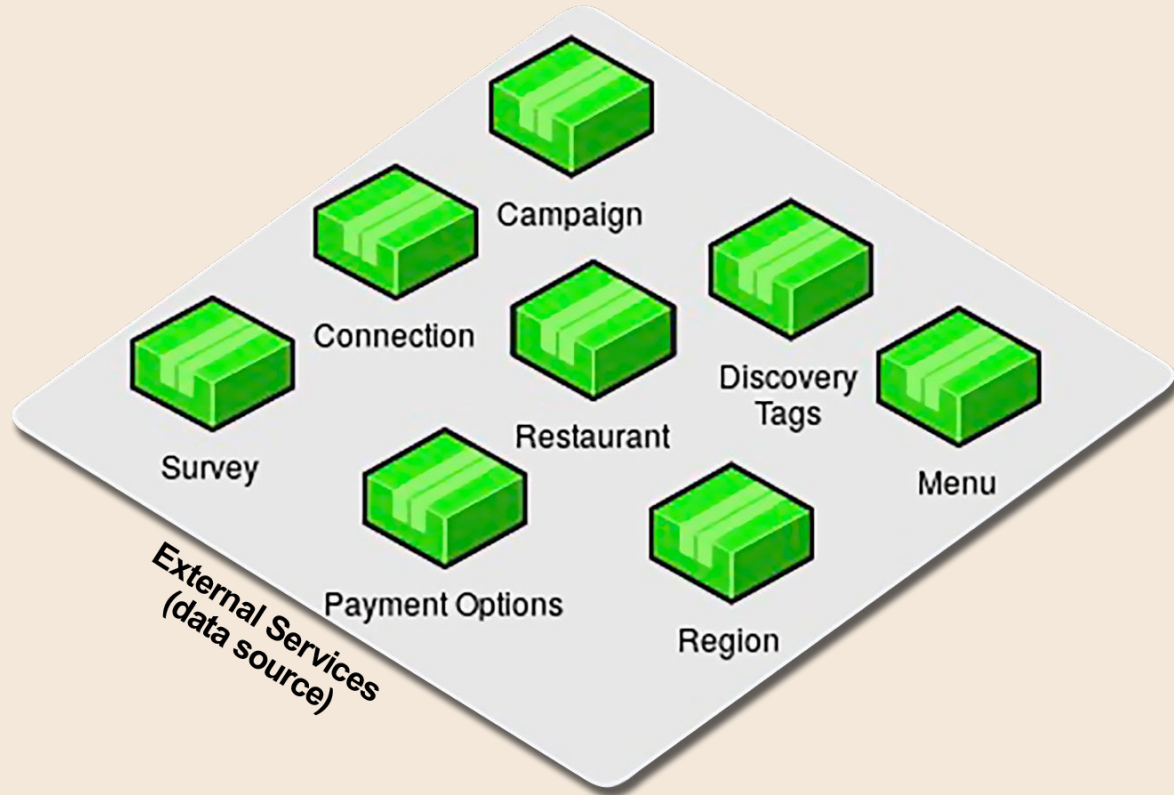


Problemas com essa solução:

- Sincronização dos dados complexa
- Difícil garantir consistência
- Tempo de sincronia alto
- Dificuldade em gerenciar caches entre as várias camadas
- Dependência de outros serviços (que poderiam não ser críticos, como o "Location Service"), dificulta manutenção
- Escalabilidade

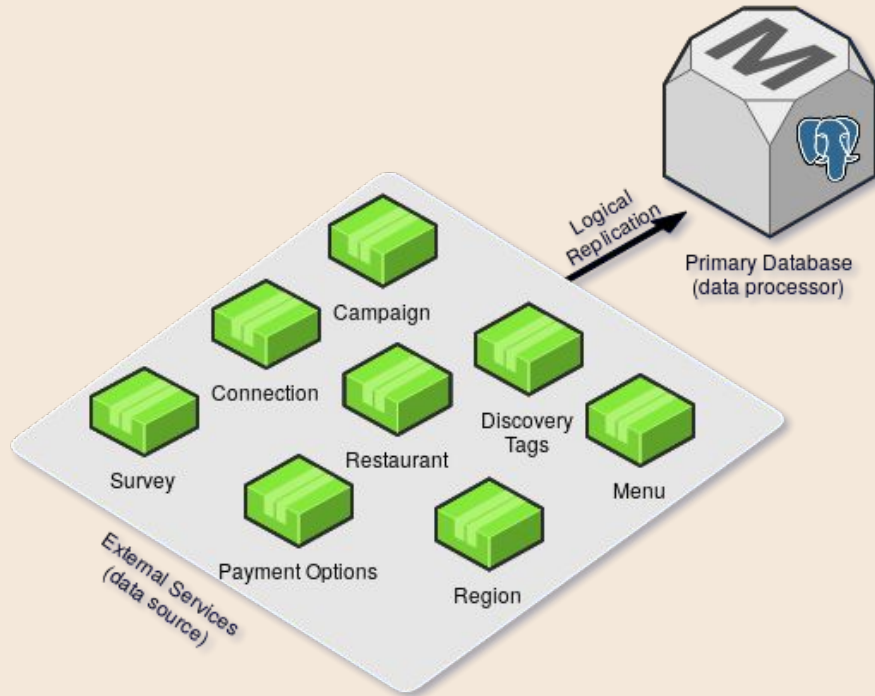
Solução

Origem dos dados



Extração dos dados

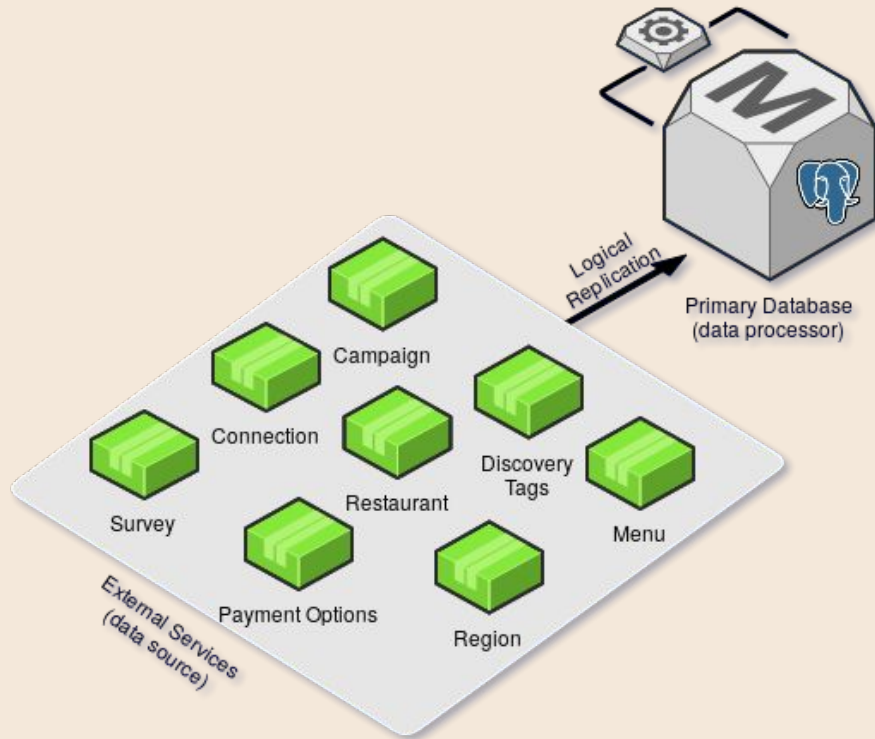
Replicamos cada uma das bases PostgreSQL para uma banco central. Cada um usando um schema, usando replicação lógica.



- Esse banco central tem uma carga maior de dados I/O
- Não é acessado por nenhuma aplicação

Processamento dos dados

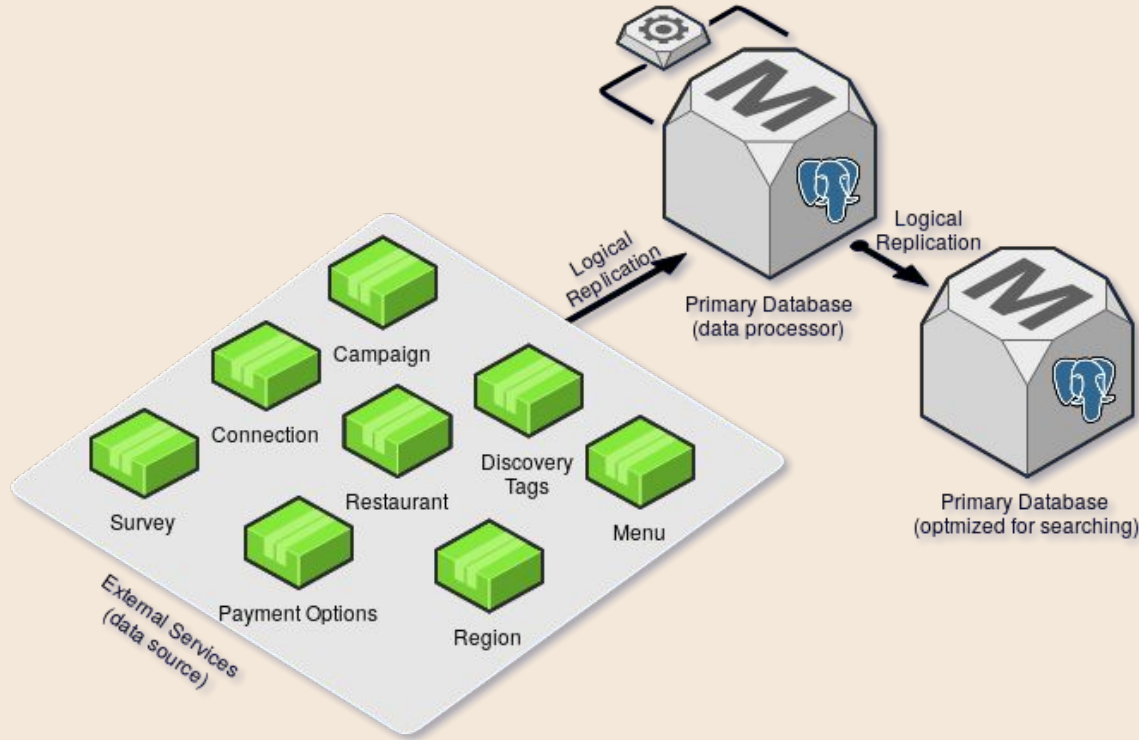
Cada tabela replicada tem um trigger, que processa os dados alimentando um novo schema otimizado para buscas



- Essas triggers não mudam o comportamento das aplicações

Replicando o schema otimizado

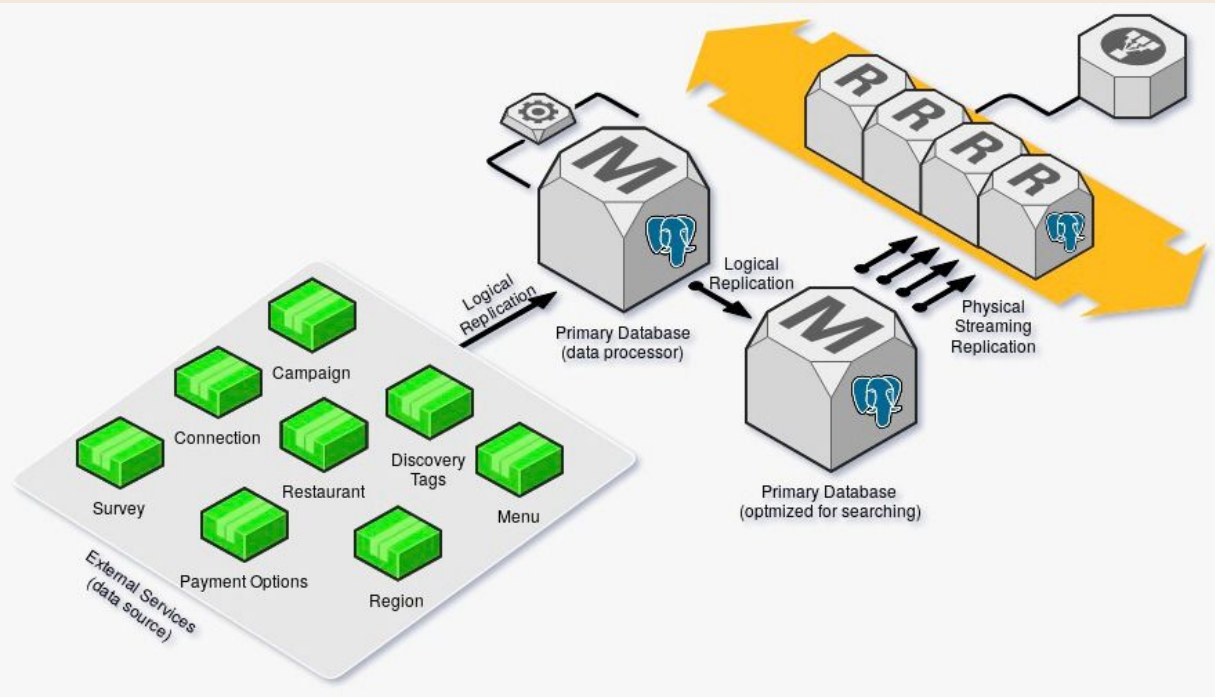
Usando replicação lógica, replicamos apenas schema otimizado para uma nova instância do PostgreSQL



- Esse banco fica “pequeno”
- Ele foi pensando apenas para leitura

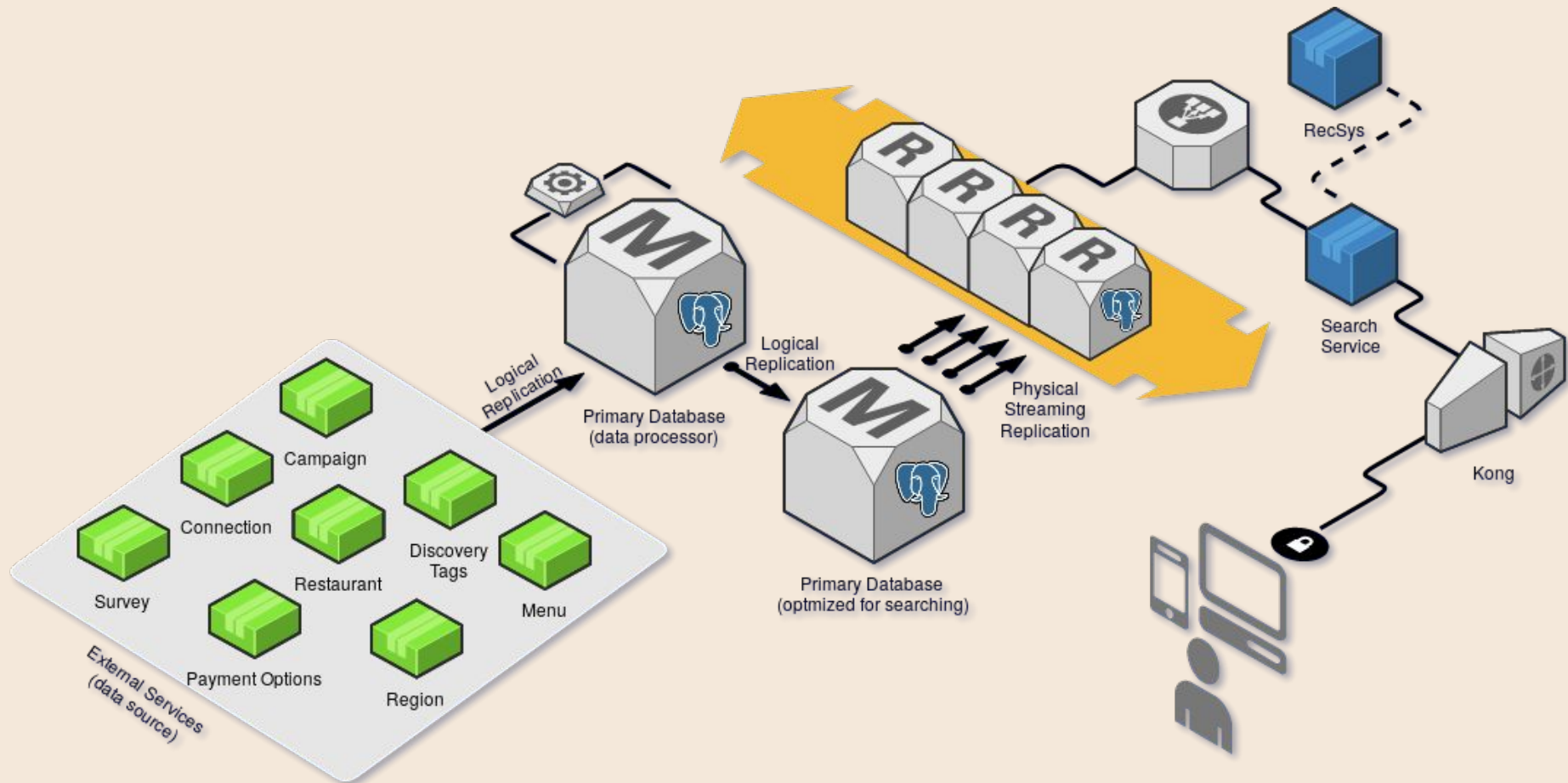
Escalando leitura

Essa instância fica num grupo de auto scaling.

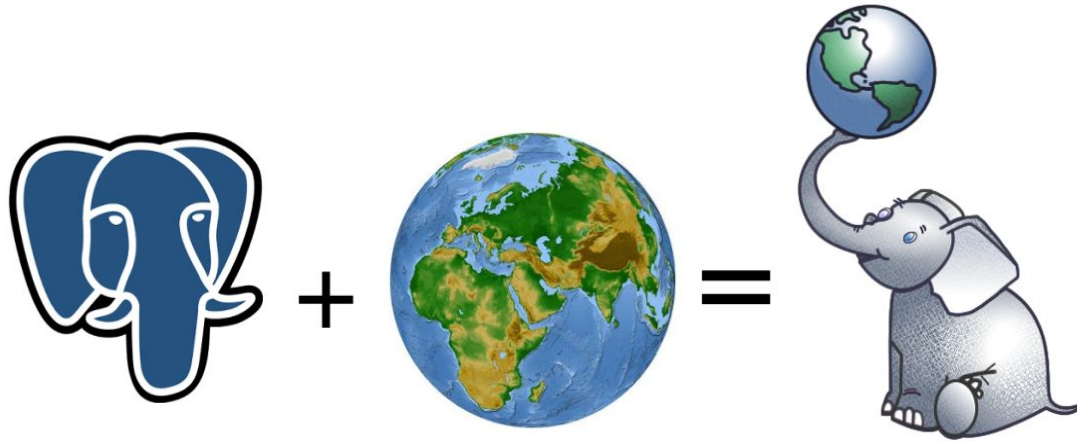


- A quantidade de instâncias depende do uso, é **elástico**
- Os dados são otimizados para leitura e pequenos para escalar rápido

Arquitetura atual



Toda busca, listagem usa lat/lon



Em nosso teste de estresse

No pico de nosso horário de teste de estresse, essa estrutura de ASG chega a **escalar ~50 máquinas c5.9xlarge (72GB de RAM cada)**



PostgreSQL

X 50

Cada réplica executa mais de 600 consultas por segundo

O que melhorou?

- Zero problemas de consistência
- Tempo de sincronia
- Mais extensível
- Fácil achar problemas
- PG é mais fácil de gerenciar que um cluster de ElasticSearch
- **Escalabilidade**
- Zero dependência dos serviços. Podem fazer deploy a vontade





Obrigado!

Lucas Viecelli

lucas.viecelli@ifood.com.br

telefone 49-99194-4888