



Dê um UP na sua aplicação

# <https://about.me/esdrasbb>

Um pouco sobre mim:

- Baiano, mas sem sotaque
- Voluntário na ABPQ
- Desenvolvedor web a mais de 10 anos
- Full stack developer
- Moviliano



A **Movile** é a empresa por trás das apps que fazem sua vida mais fácil!

The logo for sympla, featuring the word "sympla" in a blue, lowercase, sans-serif font.



The logo for Leiturinha, featuring the word "Leiturinha" in a colorful, lowercase, sans-serif font, with each letter in a different color.



Vivo Meditação

The logo for RAPIDO, featuring the word "RAPIDO" in a bold, grey, uppercase, sans-serif font, with the "O" stylized as a yellow hourglass, and the website "rapiddo.com.br" in a smaller, lowercase font below it.

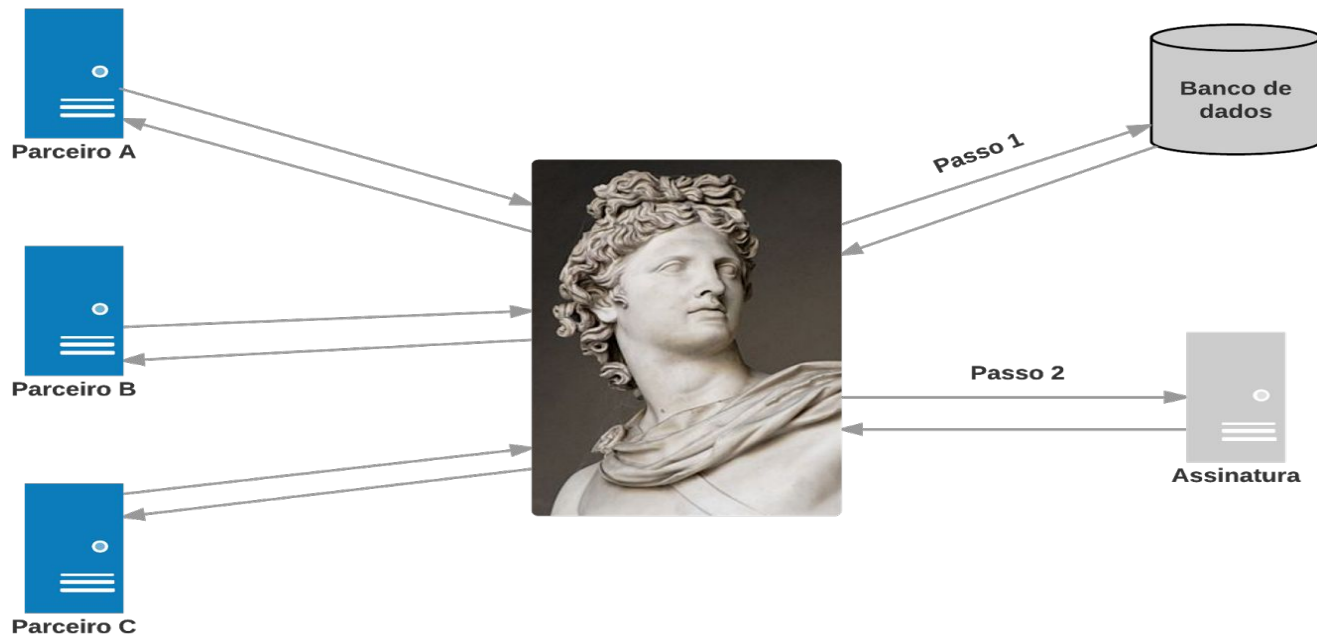
# Agenda

## Objetivos

- Apresentar um cenário real onde foi aplicada a solução do hazelcast
- Alguns detalhes e configurações
- Na prática...
  - Como adicionar o hazelcast na sua aplicação
  - Rápida análise sobre o uso em uma aplicação web.
- Considerações finais

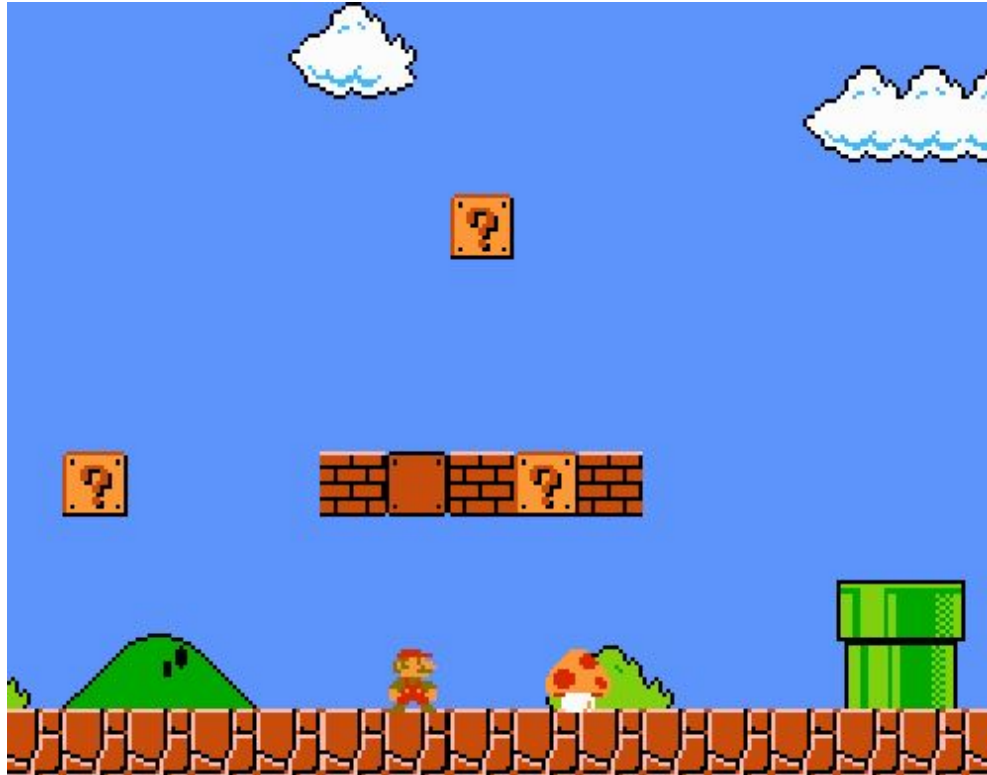
# Senta que lá vem a história...

Primeira versão do Apolo

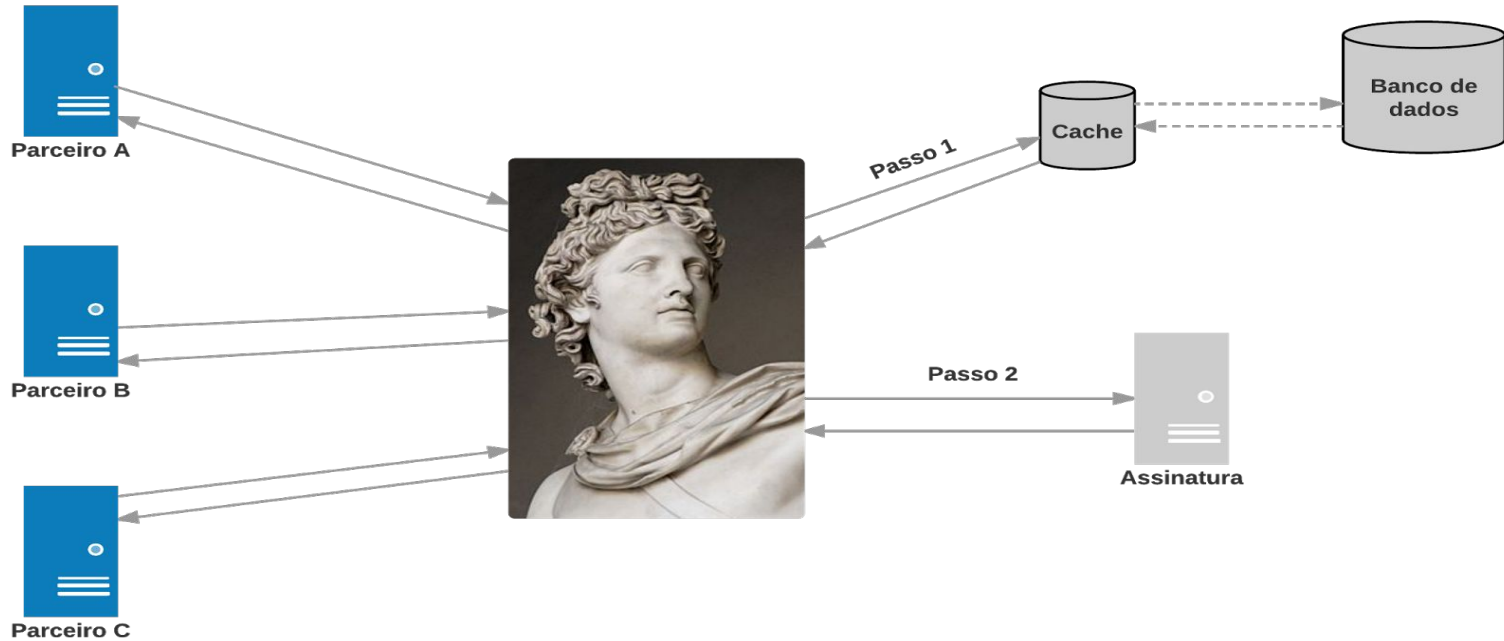


<http://apolo.com.br/assina?serviceName=Playkids&Operadora=Oi&telefone=5585912341234>

Vamos dar um UP!!!



# Segunda versão do Apolo



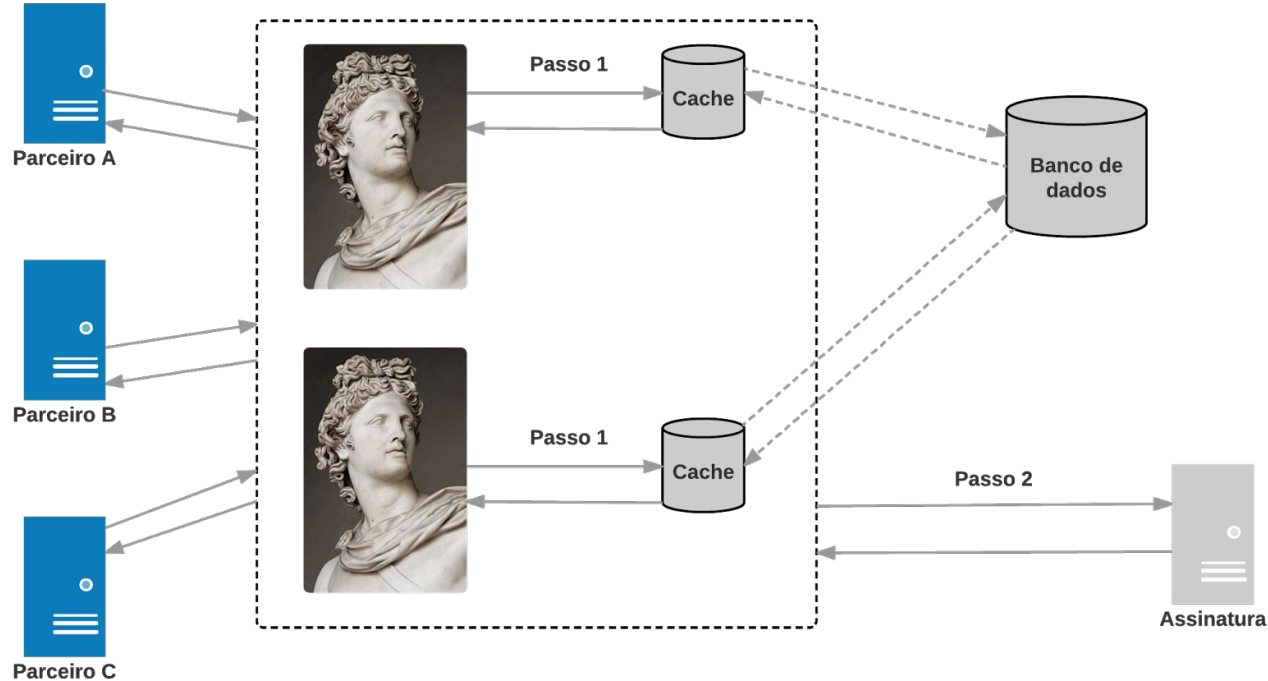
<http://apolo.com.br/assina?serviceName=Playkids&Operadora=Oi&telefone=5585912341234>

Vamos dar um UP!!!





# Terceira versão do Apolo

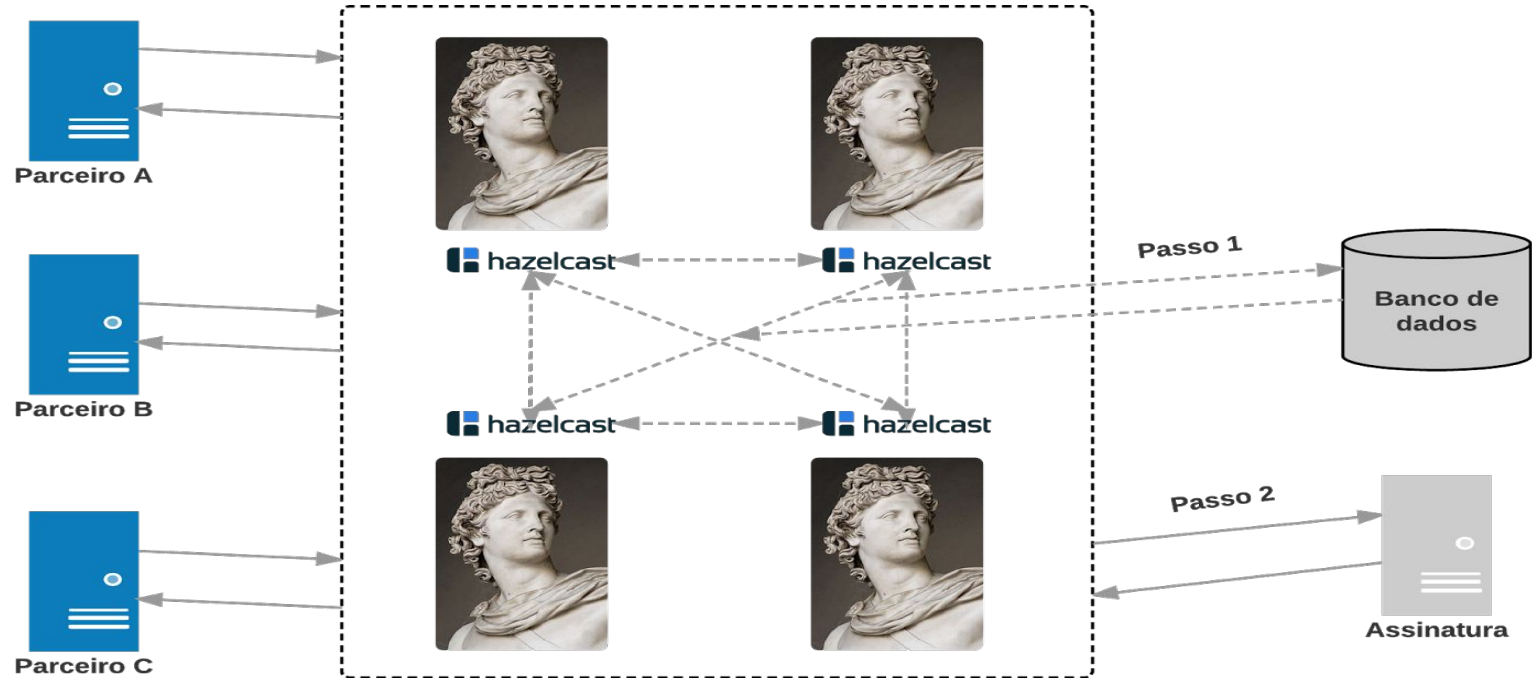


<http://apolo.com.br/assina?serviceName=Playkids&Operadora=Oi&telefone=5585912341234>

Vamos dar um UP!!!



# Versão final



<http://apolo.com.br/assina?serviceName=Playkids&Operadora=Oi&telefone=5585912341234>

# Hazelcast em detalhes...

O que é?

*Hazelcast is an open source in-memory data grid based on Java.*

Quem usa?



# Mais detalhes do Hazelcast

- Ajustes finos na configuração
  - Tamanho máximo de registros no cache (max-size)
  - Política de exclusão de registros (eviction-policy)
    - LRU: Least Recently Used - Menos recentemente usado
    - LFU: Least Frequently Used - Menos frequentemente usado
    - NONE: Ignora a configuração de tamanho máximo
  - Tempo de vida (time-to-live-seconds) de cada elemento dentro do cache
  - Tempo máximo esperando (max-idle-seconds)
- Generic Session Replication (Tomcat/Jetty/Generic)
- JMX
- Diversos Clients: Java, .NET, C++, Python, Node.js, Scala e Clojure
- Collections: Sets, Lists, Maps

# Na prática...

Como adicionar ao seu projeto?

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>1.5.5.RELEASE</version>
<relativePath/>
</parent>
...
<dependency>
<groupId>com.hazelcast</groupId>
<artifactId>hazelcast</artifactId>
<!-- Version is managed by Spring Boot starter -->
</dependency>
```

```
@SpringBootApplication
@EnableCaching
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

```
@Configuration
public class HazelcastConfiguration {

    @Bean
    public Config hazelCastConfig() {
        return new Config()
            .setInstanceName("hazelcast-instance")
            .addMapConfig(
                new MapConfig()
                    .setName("cidade")
                    .setEvictionPolicy(EvictionPolicy.LRU)
                    .setTimeToLiveSeconds(300))
            .addMapConfig(
                new MapConfig()
            )
    }
}
```

# Na prática...

Aplicação didática em spring boot + hazelcast + postgres para consultar estados e cidades.

Endpoints:

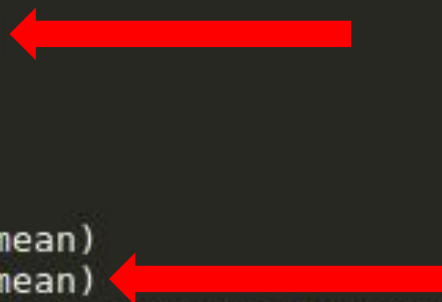
- /estado/all → todos os estados cadastrados
- /cidade/{siglaEstado} → lista de cidade pela sigla do estado.
- /clear → limpa todos os dados do cache

# Na prática...

Aplicação configurada para usar somente o **Postgres** (RDS na Amazon)

```
$> ab -n 100 -c 1 http://localhost:8080/estado/all
```

```
Concurrency Level:      1
Time taken for tests:    67.661 seconds
Complete requests:      100
Failed requests:         0
Total transferred:      126700 bytes
HTML transferred:       114800 bytes
Requests per second:    1.48 [#/sec] (mean)
Time per request:       676.608 [ms] (mean)
Time per request:       676.608 [ms] (mean, across all concurrent requests)
Transfer rate:          1.83 [Kbytes/sec] received
```

Two red arrows are present in the terminal output. The first arrow points from the right to the text '67.661 seconds' in the 'Time taken for tests:' line. The second arrow points from the right to the text '676.608 [ms] (mean)' in the 'Time per request:' line.

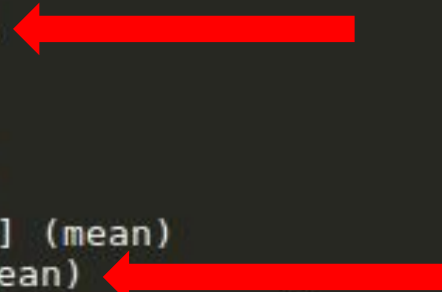


# Na prática...

Aplicação configurada para usar somente o **Hazelcast**

```
$> ab -n 100 -c 1 http://localhost:8080/estado/all
```

```
Concurrency Level:      1
Time taken for tests:    0.821 seconds
Complete requests:      100
Failed requests:         0
Total transferred:      126700 bytes
HTML transferred:       114800 bytes
Requests per second:    121.76 [#/sec] (mean)
Time per request:       8.213 [ms] (mean)
Time per request:       8.213 [ms] (mean, across all concurrent requests)
Transfer rate:          150.66 [Kbytes/sec] received
```



# Vamos aos números...

Tempo por requisição:

- Somente banco de dados → média de 676 ms (aprox.)
- Hazelcast → média de **8 ms** (aprox.)

Se considerarmos a aplicação respondendo somente a uma requisição por vez durante 1 minuto:

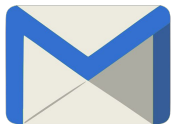
- Somente banco de dados → 89 respostas
- Hazelcast → **7.500 respostas**

# Considerações finais

- Comparação rápida com outras soluções:
  - Redis
  - Memcached
- Simples e fácil configuração
- Diversas features e ajustes finos
- Solução para cache distribuído, mas cuidado com o tamanho do grid

# MUITO OBRIGADO!!!

Contato:



*esdras.barreto@gmail.com*

Códigos e slide da apresentação:



*<https://github.com/esdrasbb/javou11>*