# MS-TRICK: ARQUITETURA DE MICROSERVICES APLICADA INTEGRADAMENTE COM APLICATIVO MÓVEL DE MANUTENÇÃO

Aluno: Ariel Rai Rodrigues

Orientadora: Simone Erbs da Costa

FURB
UNIVERSIDADE DE BLUMENAU

# Roteiro

- Introdução
- Objetivos
- Fundamentação
- Trabalhos Correlatos
- Requisitos
- Especificação
- Implementação
- Operacionalidade
- Avaliação e usabilidade
- Conclusões
- Demonstração

# Introdução

- Evolução constante
- Rápidas mudanças
- Necessidades dos clientes
- Arquitetura de microserviços
- Escalabilidade e disponibilidade

# Objetivos

- **Geral**: Aplicar uma arquitetura de microserviços no estudo de caso da empresa Pública Tecnologia

- **Específicos**
  - Dividir os módulos da aplicação para criar os microserviços;
  - Utilizar de integração contínua para facilitação na liberação de recursos;
  - Implementar uma interface móvel para manutenção e visualização dos microserviços.

# Fundamentação Teórica

- Sistema Atual – Monolito (Pública Tecnologia)
  - Sistema composto por vários módulos fortemente acoplados
- Arquitetura de software
  - Especificar, documentar, visualizar componentes de maneira abstrata
- Usabilidade de aplicativos móveis
  - Heurísticas de Nielsen
- Servidores de aplicação
  - Spring Boot

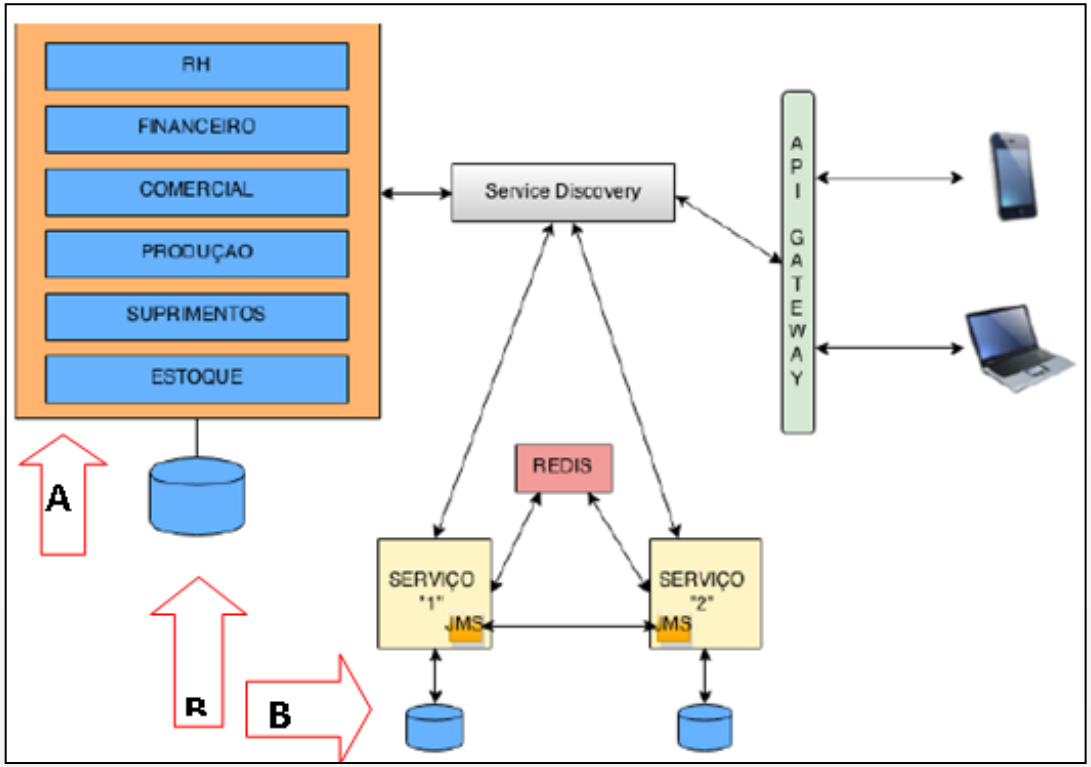
# Trabalhos Correlatos

- Netflix (Java)
- Adoção da arquitetura microserviços em aplicações corporativas (Java)
- Integrando microserviços em uma aplicação web (PHP)

# Netflix (Java)

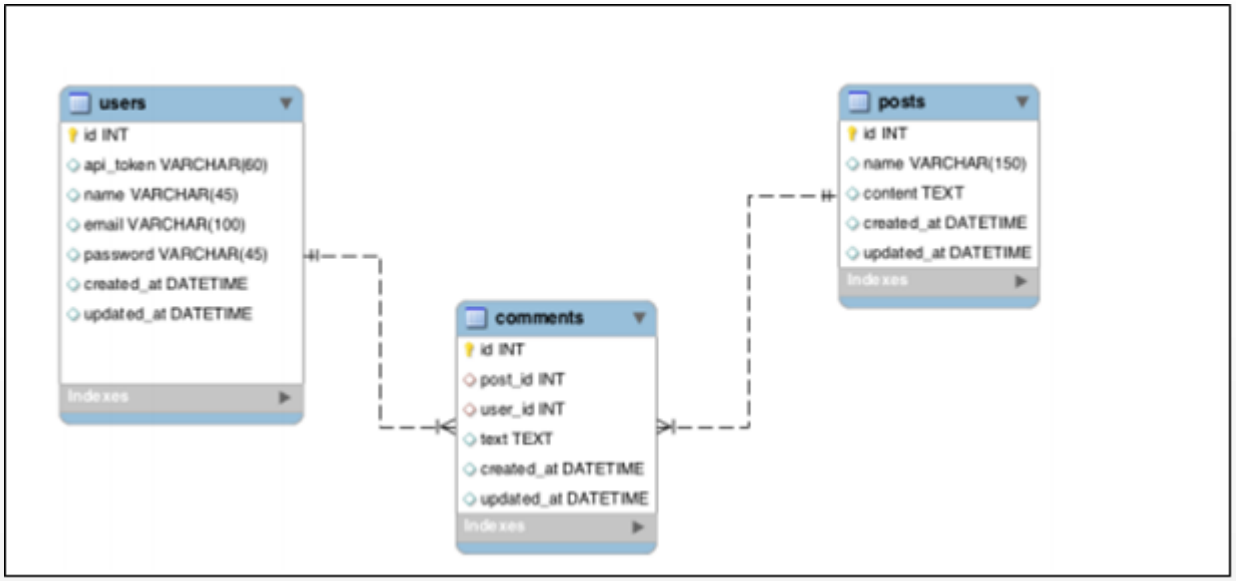

- Manutenção facilitada
- Integração Contínua
- Escalabilidade
- Disponibilidade
- Modularidade
- Ferramentas para comunidade

# Adoção da arquitetura microserviços em aplicações corporativas (Java)




- Manutenção facilitada
- Escalabilidade
- Disponibilidade
- Modularidade

# Integrando microserviços em uma aplicação web (PHP)




- Manutenção facilitada
- Métricas exclusivas.

# Requisitos funcionais

- Integração Contínua deve permitir:
  - Deploy;
  - Compilação;
  - Avaliação do código.
- Arquitetura deve permitir:
  - Execução de múltiplas instâncias.

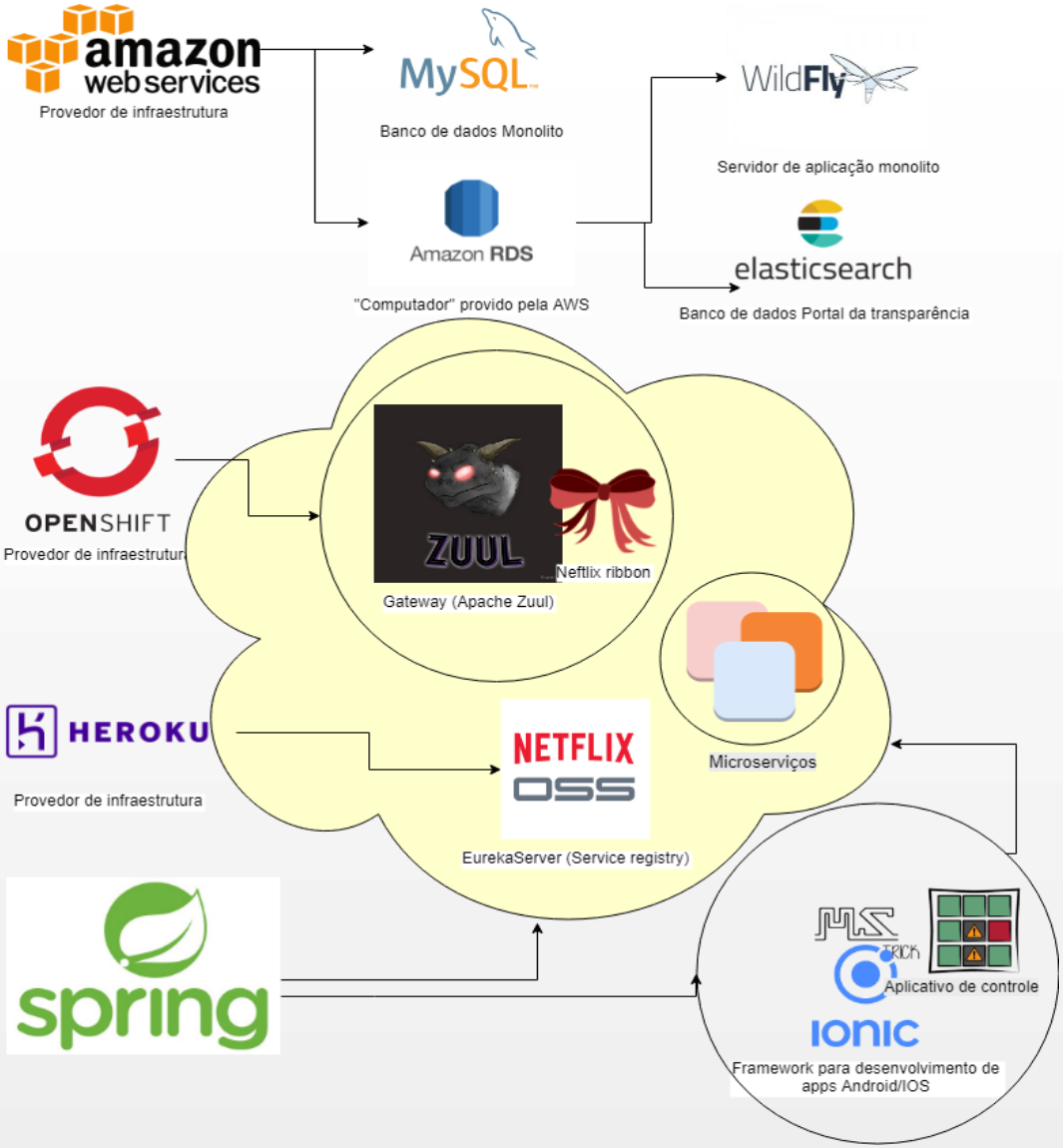
# Requisitos funcionais

- Aplicativo deve permitir:
  - Realização de *login*
  - Reiniciar instância de microserviço
  - Desligar instância de microserviço
  - Habilitar/Desabilitar notificações
  - Recebimento de notificações
  - Ao *backend* o envio de notificações
- Visualizar:
  - instâncias de microserviços;
  - métricas de microserviços;
  - variáveis de ambiente do microserviço
  - *endpoints* do microserviço;
  - requisições feitas aos microserviços

# Requisitos não Funcionais

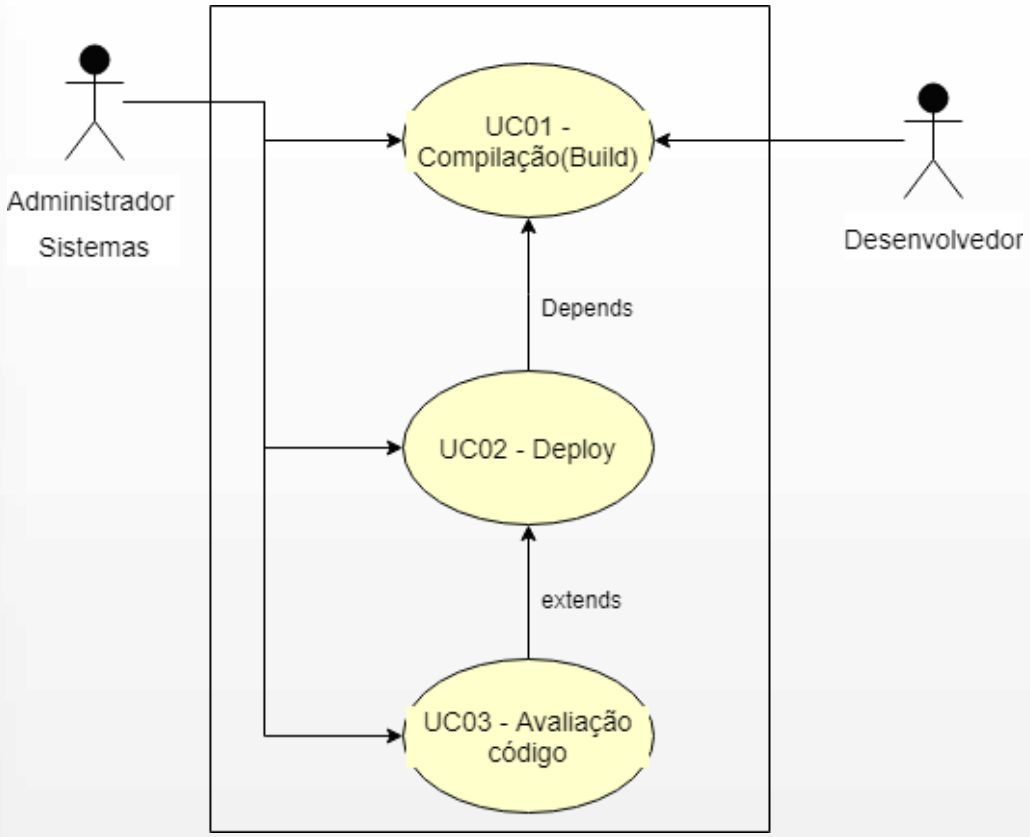
- Integração Contínua deve utilizar:
  - Gitlab
- Arquitetura deve Utilizar:
  - Tecnologias disponibilizadas pelo Netflix: Hystrix, Ribbon, Zuul.
- Microserviços devem:
  - Ser implementados utilizando Spring Boot
- Aplicativo deve:
  - Ter a interface implementada utilizando Ionic
  - Ter o b*ackend* em Spring Boot

# Diagrama de tecnologias



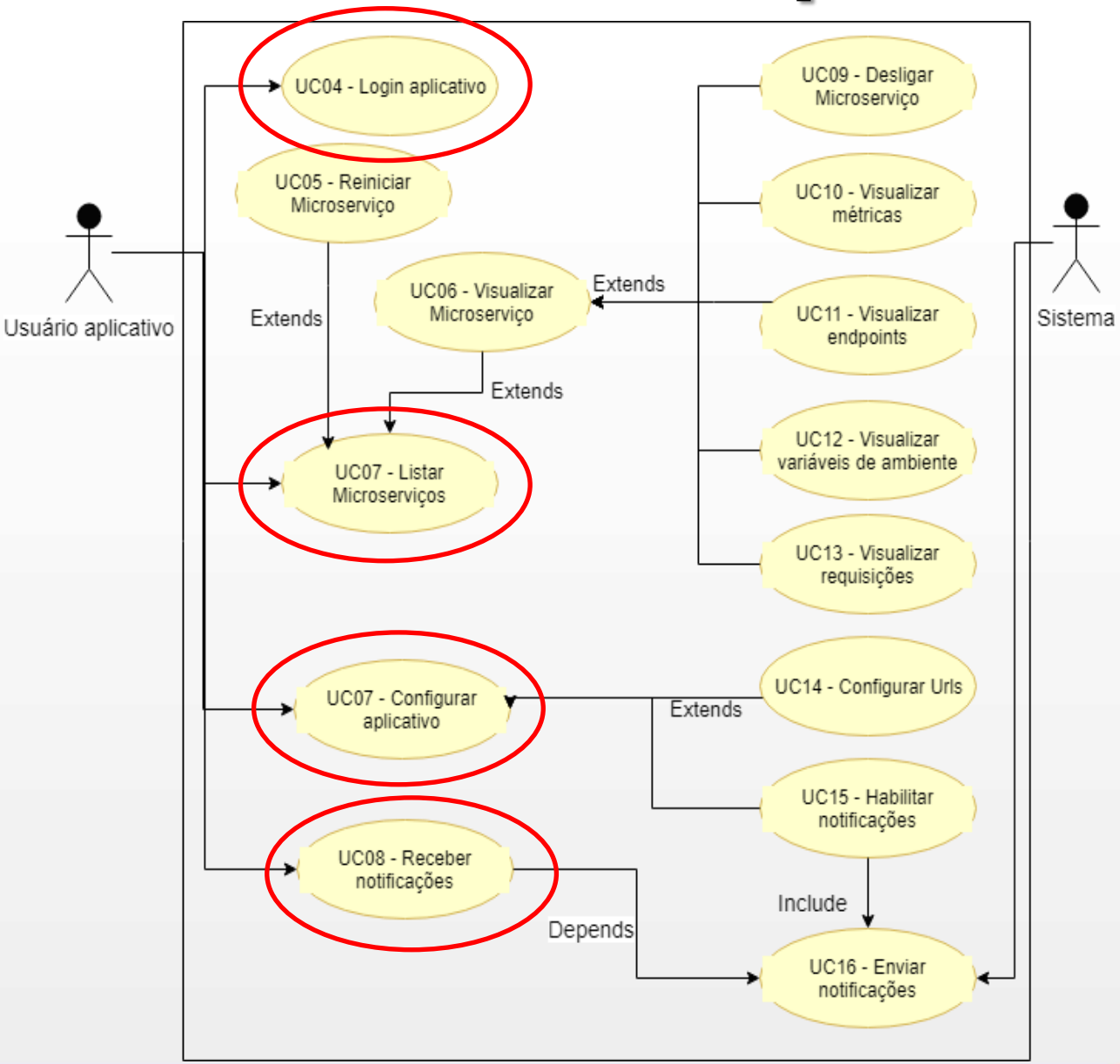

- Provedores de infraestrutura
- Spring
- Hystrix
- Ribbon
- Eureka
- Ionic

# Fluxo de liberação



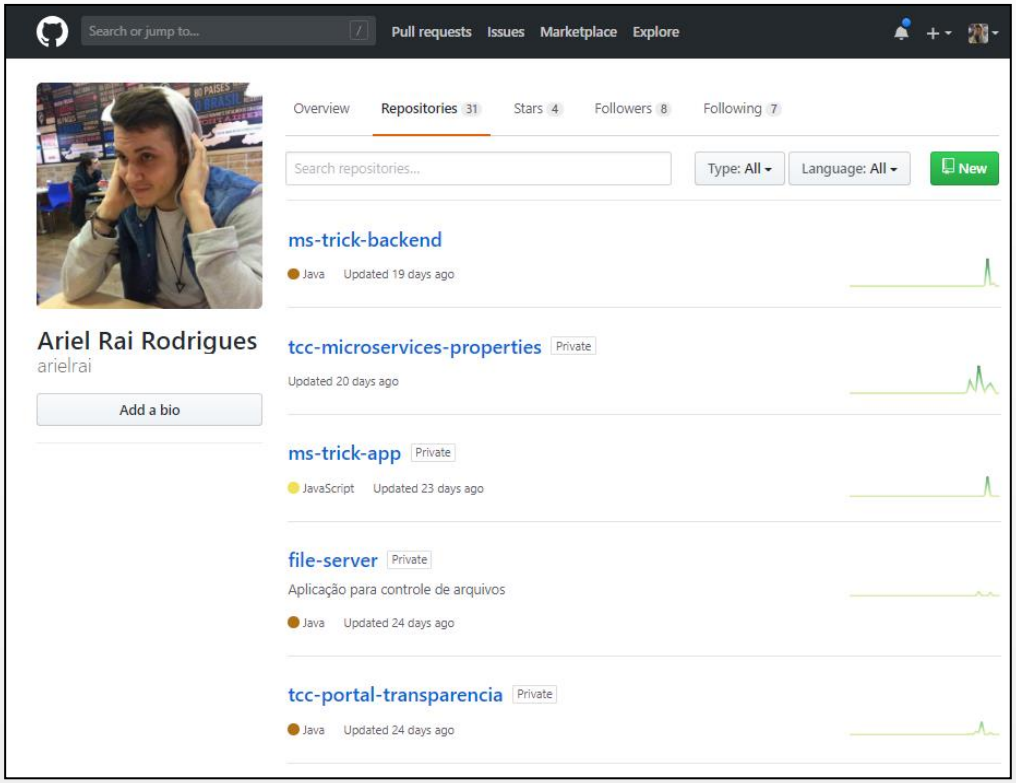

- Fluxo Manual
  - Administrador
- Fluxo Automático
  - Desenvolvedor
- Avaliação de código
  - SonarQube

# Casos de uso aplicativo



# Implementação

- Ferramentas para desenvolvimento
  - *Backend*: Red hat Develoment Suite
  - *Frontend*: Atom




- Github
  - Armazenamento de código

# Código *Backend*

- Java
- Spring boot

15 lines (11 sloc) | 410 Bytes

```java
package com.publica;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {

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

# Código *Frontend*

app.html

ion-view | ion-content.has-header | div | div.item.item-body

```
<ion-view title="{{app.applicationName}}">
  <ion-nav-buttons side="left">
    <button menu-toggle="left" class="button button-icon icon ion-navicon"></button
  </ion-nav-buttons>
  <ion-content class="has-header" style="margin-bottom: 40px">
    <div ng-if="menu == 'home'">
      <div class="item">
        <h2><b>{{app.applicationName}}</b></h2>
        <p><u>Ativo desde: {{app.upSince | date : 'dd/MM/yyyy - H:ss'}}</u></p>
      </div>

      <div class="item item-body">
        <b>Hostname:</b> {{app.host}}<br>
        <b>Ip:</b> {{app.ipAdress}}<br>
        <b>Instance Id:</b> {{app.instanceId}}
      </div>
```

app.js | controllers.js

```
.controller('AppCtrl', function ($scope, $state, $http, URLService, $stateParams,
  $scope.getKeys = function (object) {
    return Object.keys(object);
  }
  $scope.toggleGroup = function (group) {
    group.show = !group.show;
  };
  $scope.isGroupShown = function (group) {
    return group.show;
  };
  $scope.app = JSON.parse($stateParams.app);
  $scope.menu = 'home';
  $scope.restart = function (app) {
    $ionicLoading.show();
    $http.post(URLService.buildUrl('/thirdPartyPost'), buildThirdPartyRequest(wir
      setTimeout(function () {
```

- *Ionic*

# Script integração contínua

Branch: master ▾ ms-trick-backend / .gitlab-ci.yml

arielrai no message

1 contributor

73 lines (66 sloc) | 1.41 KB

```
1  cache:
2   paths:
3   - /root/.m2/repository
4
5  stages:
6   - build
7   - deploy
8   - sonar
9
10 build:
11  stage: build
12  image: maven:3.3.9-jdk-8
13  script:
14  - mvn clean install -DskipTests
15  tags:
16  - docker
17
18 sonar:
19  stage: sonar
20  image: maven:3.3.9-jdk-8
21  script:
22  - mvn clean install -DskipTests sonar:sonar -Dsonar.organization=arielrai-github -Dso
23  tags:
24  - docker
25
26 deploy_staging:
27  stage: deploy
```

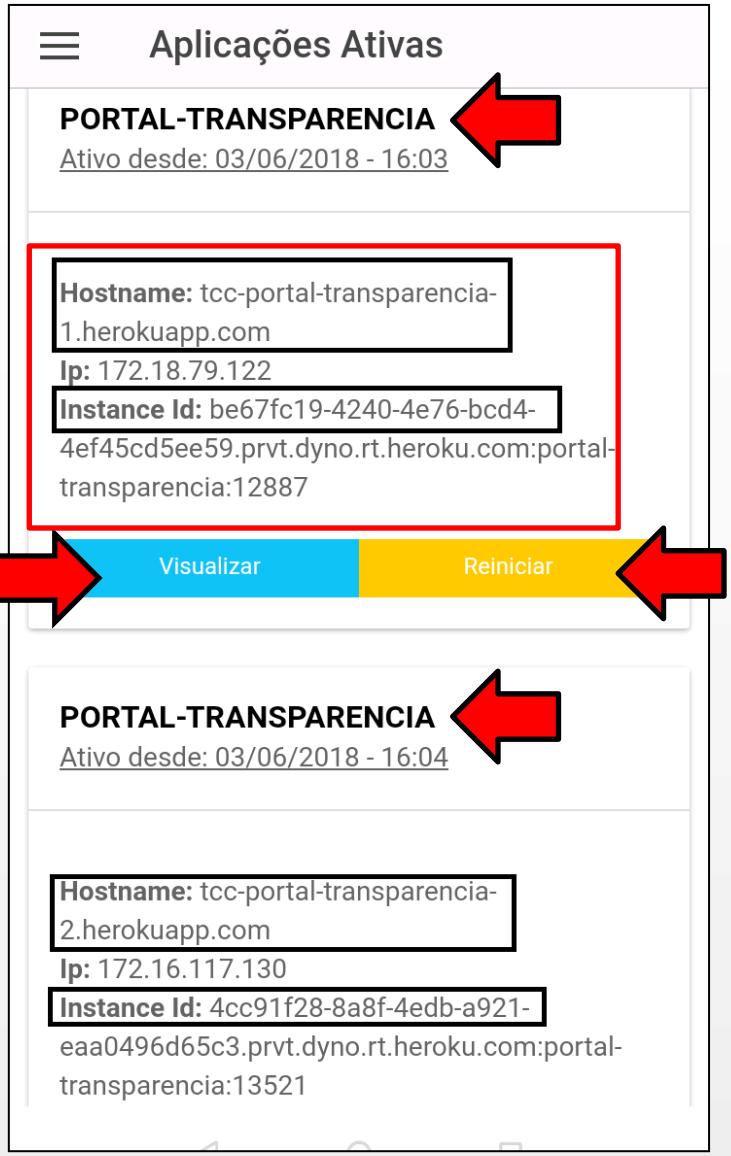
Stages

Build

Sonar (avaliação do código)

Deploy

# Operacionalidade




- Possibilidade de replicação
- Nome do microserviço
- Informações básicas sobre a instância
- Reiniciar
- Visualizar

# Resultados e Discussões

- Correlação do MS-Trick com os correlatos
- Avaliação de usabilidade
  - Três usuários especialistas da empresa
  - Google Formulários
  - Acompanhamento via Skype

# Correlação dos correlatos com MS-Trick

| Características \ Trabalhos | Tonse (2014) | Silva (2015) | São Miguel e Farina (2016) | MS-Trick |
|---|---|---|---|---|
| Escalabilidade | ✓ | ✓ | X | ✓ |
| Integração contínua | ✓ | X | X | ✓ |
| Disponibilidade | ✓ | ✓ | X | ✓ |
| Métricas exclusivas | X | X | ✓ | ✓ |
| Modularidade | ✓ | ✓ | X | ✓ |
| Linguagem de programação | Java | Java | PHP | Java |
| Melhoria no processo de manutenção | ✓ | ✓ | X | ✓ |
| Aplicativo de controle | X | X | X | ✓ |

# Avaliação


- Todos os usuários visualizaram vantagem


- Todos os usuários visualizaram o aplicativo como eficiente para seu propósito

# Avaliação de Usabilidade

# Síntese das heurísticas aferidas

- H4, H6, H9
  - Problemas irrelevantes
  - Relacionados a interface
- H10
  - Ideia de extensão

# Conclusões

- Escalabilidade e disponibilidade
  - Alcançada pela replicação de serviços
- Variedade de tecnologias
  - Microserviços permitem utilizar qualquer tecnologia
- Integração contínua
  - Facilita o processo de desenvolvimento e liberação de uma aplicação

# Contribuições

- **Aplicado**
  - Alcançou expectativas da empresa
- **Tecnológica**
  - Demonstrar aplicação da arquitetura
    - Utilização de Netflix *Stack*
    - Utilização de Gitlab
    - Utilização de Spring Boot
- **Acadêmica**

- Aplicar a arquitetura de microserviços numa aplicação existente

# Dificuldades Encontradas

- Entendimento de todas tecnologias utilizadas
  - Integração das tecnologias
- Máquina utilizada no desenvolvimento
- Demonstração da arquitetura integradamente

# Extensões

- Glossário de termos técnicos
- Dockerização
- Kubernetes
- Extensão Gitlab
- Externalização de demais módulos

# Demonstração

# MS-TRICK: ARQUITETURA DE MICROSERVIÇOS APLICADA INTEGRADAMENTE COM APLICATIVO MÓVEL DE MANUTENÇÃO

Aluno: Ariel Rai Rodrigues

Orientadora: Simone Erbs da Costa

FURB
UNIVERSIDADE DE BLUMENAU