

TECH CHALLENGE

FASE 3

ARQUITETURA E DESENVOLVIMENTO  
JAVA

# Integrantes do Grupo

César Alves Vecchio  
calvesvecchio@gmail.com

Rodrigo Henrique Knob  
r\_knob@outlook.com

Leticia Jorge dos santos  
leticiajds22@gmail.com

João Marcos Almeida Silva de Jesus  
thespacejhony@gmail.com

Guilherme Matos de Carvalho  
guilhermematos851@gmail.com

## Link do projeto no GitHub:

<https://github.com/cesarvecchio/restaurante-adjt>.

## Como executar o projeto:

Para executar o projeto, é necessário ter instalado o **Java 17**, **Gradle 7+** e **MongoDB rodando localmente na porta 27017**.

Após clonar o projeto, se estiver utilizando o IntelliJ, basta abrir o projeto e executar a classe **RestauranteAdjApplication.java**

Na pasta **resources** do projeto há um arquivo com a collection do Postman, para testar os principais endpoints do fluxo.

## Sugestão de ordem para execução dos Endpoints para teste do Fluxo principal:

- Criar **Restaurante**;
- Listar **Restaurante** por nome e/ou tipo cozinha e/ou localização;
- Criar **Reserva** por restaurante;
- Atualizar status da **Mesa** por reserva;
- Listar **Mesas** por restaurante, data da reserva, horário da reserva e status;
- Criar **Avaliação** por reserva;
- Listar **Avaliações** por restaurante;

## Como executar os testes:

Requisitos para executar os comandos a seguir:

- Ter acesso ao comando **make** (Necessário para execução de todos os comandos a seguir);
- Ter o **docker** instalado (Necessário para execução de comandos relacionados ao docker);
- Ter o **Gradle** instalado (Necessário para execução de todos os comandos a seguir);
- Ter o **Node/NPM** instalado + o pacote **allure-commandline** globalmente (Necessário para visualização dos relatórios dos testes integrados);

## Para executar os testes unitários:

```
» make unit-test
```

*make unit-test*

## Para executar os testes integrados:

```
» make integration-test
```

*make integration-test*

Obs.: Após executar os testes integrados é possível visualizar relatórios dos testes executando o comando `allure -serve ${caminhoDo Projeto}/build/allure-results`

## Para executar os testes unitários + testes integrados:

```
» make test
```

*make test*

## Para executar os testes de sistema/comportamento:

O fluxo recomendado para esse teste é de iniciar 2 contêineres docker, um executando a aplicação e outro executando um banco de dados mongodb, com a finalidade de não sujar a sua base de dados do mongodb instalado localmente em sua máquina.

Para isso vamos seguir essas 5 etapas:

### 1) Realizar o build do projeto:

```
» make build-project
```

*make build-project*

### 2) Realizar o build da imagem da aplicação:

```
» make docker-build
```

*make docker-build*

### 3) Subir os contêineres:

```
» make docker-start
```

*make docker-start*

### 4) Executar os testes de sistema/comportamento:

```
» make system-test
```

*make system-test*

### 5) Remover os contêineres e recursos que foram criados junto com eles:

```
» make docker-stop
```

*make docker-stop*

**Obs.:** Após executar os testes de sistema/comportamento é possível visualizar relatórios dos testes abrindo o arquivos .html localizado dentro da pasta `${caminhoDoProjeto}/build/cucumber-reports`

## Para executar os testes de performance:

O fluxo recomendado para os testes de performance seguem na mesma linha dos testes de sistema, iniciar 2 contêineres docker, um executando a aplicação e outro executando um banco de dados mongodb.

Para isso vamos seguir essas 8 etapas:

### 1) Realizar o build do projeto:

```
» make build-project
```

*make build-project*

### 2) Realizar o build da imagem da aplicação:

```
» make docker-build
```

*make docker-build*

### 3) Subir os contêineres:

```
» make docker-start
```

*make docker-start*

### 4) Executar os testes de performance de restaurante:

```
» make performance-restaurant
```

*make performance-restaurant*

### 5) Executar os testes de performance de reserva:

```
» make performance-reserva
```

*make performance-reserva*

### 6) Executar os testes de performance de mesa:

```
» make performance-mesa
```

*make performance-mesa*

### 7) Executar os testes de performance de avaliação:

```
» make performance-avaliacao
```

*make performance-avaliacao*

**Obs.: Caso queira executar todos os testes de performance utilize o seguinte comando:**

```
» make performance-test
```

*make performance-test*

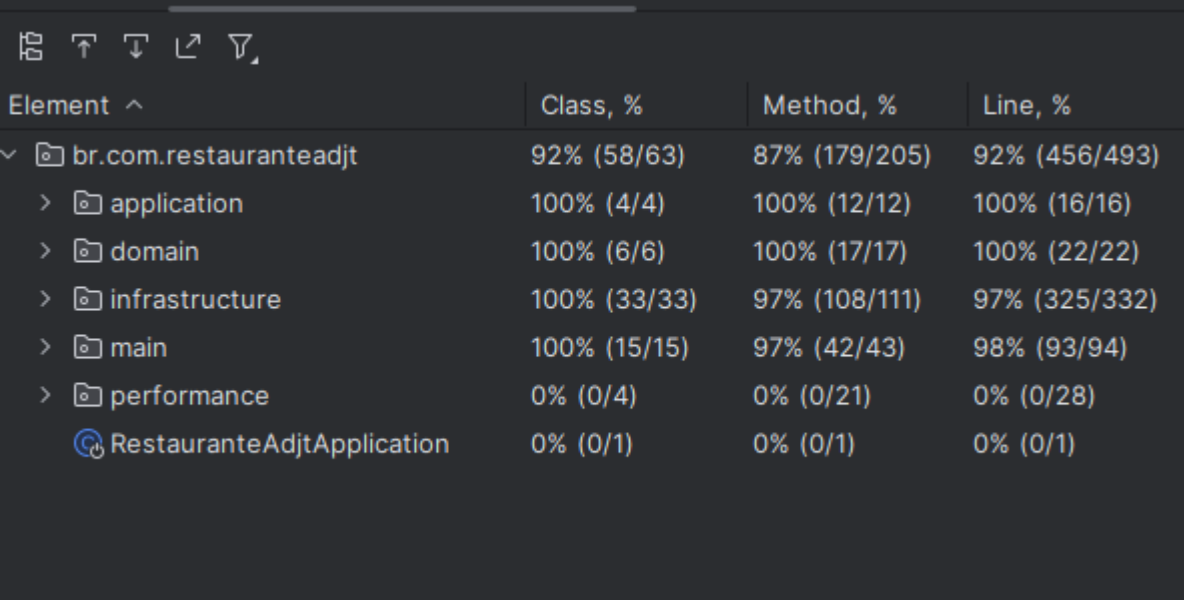
### 8) Remover os contêineres e recursos que foram criados junto com eles:

```
» make docker-stop
```








*make docker-stop*

Obs.: É possível visualizar relatórios dos testes de performance abrindo os arquivos .html nas pastas `${caminhoDoProjeto}/build/reports/gatling`

## Cobertura final do código:



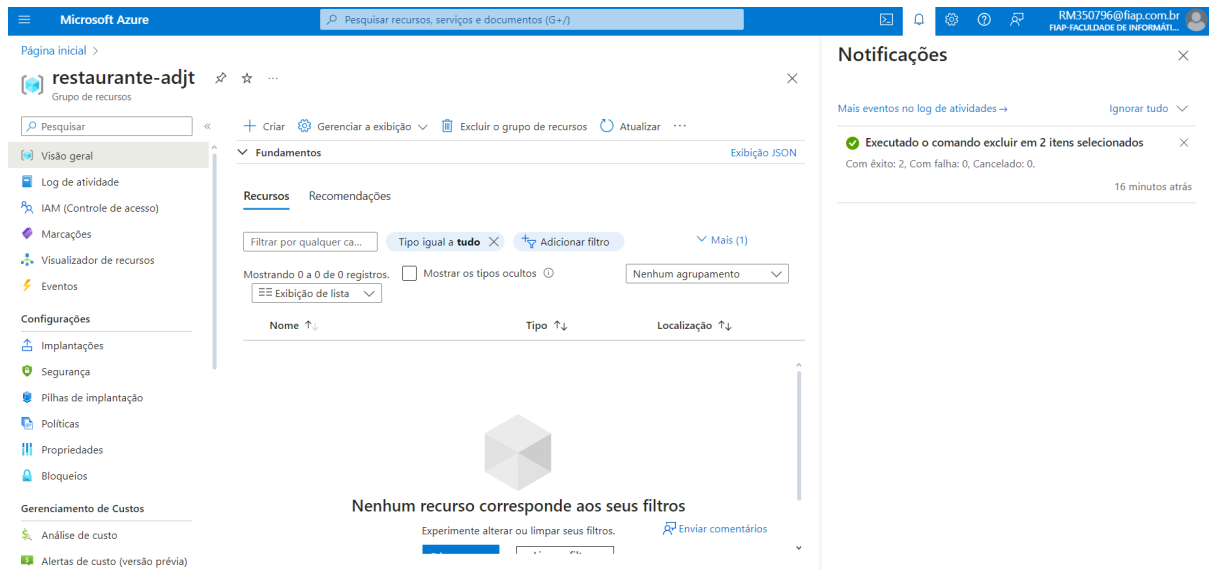
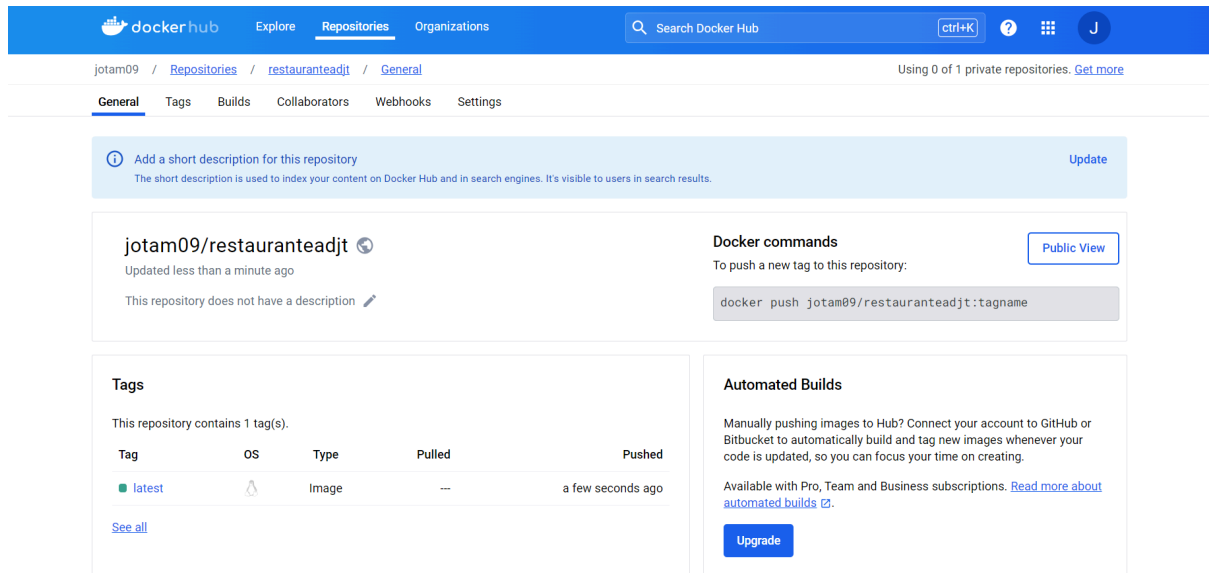
The screenshot shows the IntelliJ IDEA interface with a table of code coverage data. The table has four columns: 'Element', 'Class, %', 'Method, %', and 'Line, %'. The data is organized hierarchically, starting with the project root 'br.com.restauranteadjt' and then listing sub-packages like 'application', 'domain', 'infrastructure', 'main', 'performance', and a specific class 'RestauranteAdjApplication'.

Element ^	Class, %	Method, %	Line, %
✓  br.com.restauranteadjt	92% (58/63)	87% (179/205)	92% (456/493)
>  application	100% (4/4)	100% (12/12)	100% (16/16)
>  domain	100% (6/6)	100% (17/17)	100% (22/22)
>  infrastructure	100% (33/33)	97% (108/111)	97% (325/332)
>  main	100% (15/15)	97% (42/43)	98% (93/94)
>  performance	0% (0/4)	0% (0/21)	0% (0/28)
 RestauranteAdjApplication	0% (0/1)	0% (0/1)	0% (0/1)

## Deploy do código na Azure:

Como temos acesso a conta de estudante com créditos para utilização dos serviços, optamos por utilizar a Azure para fazer o deploy da nossa aplicação.

Já com a imagem docker da nossa aplicação no DockerHub (jotam09/restauranteadjt), o Azure CLI (Interface de Linha de Comando da Azure) instalado e o Grupo de Recursos restaurante-adjt criado.



Rodamos os seguintes comandos para fazer a criação de 2 contêineres, um para a aplicação e outro para o banco de dados MongoDB:



```
az container create \
--resource-group restaurante-adjt \
--name mongodb-restaurante \
--image mongo:7.0 \
--dns-name-label mongodb-restaurante \
--ports 27017 \
--environment-variables MONGO_INITDB_ROOT_USERNAME=mongoadmin MONGO_INITDB_ROOT_PASSWORD=mongopassword
```

Executando o comando no Terminal:

```
joao@DESKTOP-0MFQ4QS: /mnt/c/Users/João_Marcos/Documents/restaurante-adjt$ az container create \
--resource-group restaurante-adjt \
--name mongodb-restaurante \
--image mongo:7.0 \
--dns-name-label mongodb-restaurante \
--ports 27017 \
--environment-variables MONGO_INITDB_ROOT_USERNAME=mongoadmin MONGO_INITDB_ROOT_PASSWORD=mongopassword
{
  "confidentialComputeProperties": null,
  "containers": [
    {
      "command": null,
      "environmentVariables": [
        {
          "name": "MONGO_INITDB_ROOT_USERNAME",
          "secureValue": null,
          "value": "mongoadmin"
        },
        {
          "name": "MONGO_INITDB_ROOT_PASSWORD",
          "secureValue": null,
          "value": "mongopassword"
        }
      ],
      "image": "mongo:7.0",
      "instanceView": {
        "currentState": {
          "detailStatus": "",
          "exitCode": null,
          "finishTime": null,
          "startTime": "2024-03-26T01:57:37.925000+00:00",
          "state": "Running"
        }
      },
      "events": [
        {
          "count": 1,
          "firstTimestamp": "2024-03-26T01:57:11+00:00",
          "lastTimestamp": "2024-03-26T01:57:11+00:00",

```

Portal Azure após executar o comando acima:

Microsoft Azure

Página inicial >

### restaurante-adjt

Grupo de recursos

Log de atividade  
IAM (Controle de acesso)  
Marcações  
Visualizador de recursos  
Eventos

Configurações  
Implantações  
Segurança  
Pilhas de implantação  
Políticas  
Propriedades  
Bloqueios

Gerenciamento de Custos  
Análise de custo  
Alertas de custo (versão prévia)

Recursos

Fundamentos

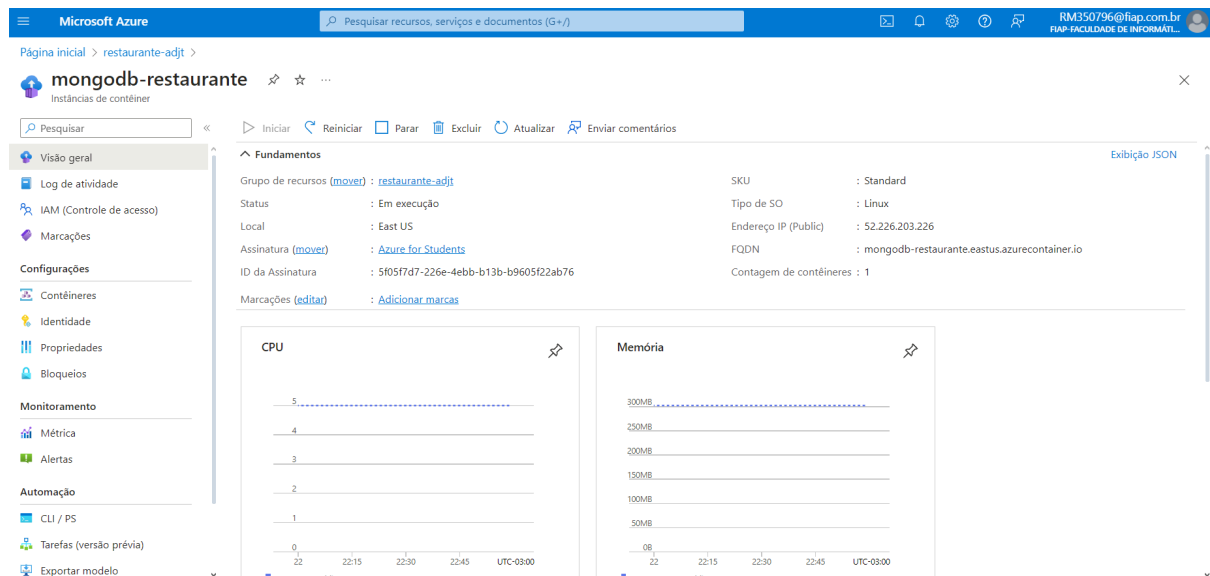
Exibição JSON

Mostrando 1 a 1 de 1 registros.

Nome	Tipo	Localização
mongodb-restaurante	Instâncias de contêiner	East US

< Anterior    Página 1 de 1    Próximo >

Enviar comentários



```
az container create \
--resource-group restaurante-adjt \
--name restaurante-api \
--dns-name-label restaurante-api \
--ports 8080 \
--image jotam09/restauranteadjt \
--restart-policy OnFailure \
--environment-variables 'MONGO_URI'='mongodb://mongoadmin:mongopassword@mongodb-restaurante.eastus.azurecontainer.io:27017/restaurante?authSource=admin'
```

Executando o comando no Terminal:

```
joao@DESKTOP-0MFQ4QS: /mnt/c/Users/João Marcos/Documents/restaurante-adjt$ az container create \
--resource-group restaurante-adjt \
--name restaurante-api \
--dns-name-label restaurante-api \
--ports 8080 \
--image jotam09/restauranteadjt \
--restart-policy OnFailure \
--environment-variables 'MONGO_URI'='mongodb://mongoadmin:mongopassword@mongodb-restaurante.eastus.azurecontainer.io:27017/restaurante?authSource=admin'
{
  "confidentialComputeProperties": null,
  "containers": [
    {
      "command": null,
      "environmentVariables": [
        {
          "name": "MONGO_URI",
          "secureValue": null,
          "value": "mongodb://mongoadmin:mongopassword@mongodb-restaurante.eastus.azurecontainer.io:27017/restaurante?authSource=admin"
        }
      ],
      "image": "jotam09/restauranteadjt",
      "instanceView": {
        "currentState": {
          "detailStatus": "",
          "exitCode": null,
          "finishTime": null,
          "startTime": "2024-03-26T02:01:21.281000+00:00",
          "state": "Running"
        },
        "events": [
          {
            "count": 1,
            "firstTimestamp": "2024-03-26T02:01:00+00:00",
            "lastTimestamp": "2024-03-26T02:01:00+00:00",
            "message": "pulling image \jotam09/restauranteadjt@sha256:c638ecbadf557c13de12beac81044e607a79f438bd4f99bd38ed0698b2e661e\"",
            "name": "Pulling",
            "type": "Normal"
          }
        ]
      }
    }
  ]
}
```

Portal Azure após executar o comando acima:

Microsoft Azure

Pesquisar recursos, serviços e documentos (G+)

Página inicial > restaurante-adjt

Grupo de recursos

Visão geral

- Log de atividade
- IAM (Controle de acesso)
- Marcações
- Visualizador de recursos
- Eventos

Configurações

- Implantações
- Segurança
- Pilhas de implantação
- Políticas
- Propriedades
- Bloqueios

Gerenciamento de Custos

- Análise de custo
- Alertas de custo (versão prévia)

Fundamentos

Recursos

Filtrar por qualquer ca...

Tipo igual a **tudo**

Localização igual a **tudo**

Adicionar filtro

Mostrando 1 a 2 de 2 registros. ☐ Mostrar os tipos ocultos

Nenhum agrupamento

Exibição de lista

Nome	Tipo	Localização
mongodb-restaurante	Instâncias de contêiner	East US
restaurante-api	Instâncias de contêiner	East US

< Anterior

Página 1 de 1

Próximo >

Enviar comentários

Microsoft Azure

Pesquisar recursos, serviços e documentos (G+)

Página inicial > restaurante-api

Instâncias de contêiner

Visão geral

- Log de atividade
- IAM (Controle de acesso)
- Marcações
- Contêineres
- Identidade
- Propriedades
- Bloqueios

Monitoramento

- Métrica
- Alertas

Automação

- CLI / PS
- Tarefas (versão prévia)
- Exportar modelo

Fundamentos

Grupo de recursos (mover) : restaurante-adjt

Status : Em execução

Local : East US

Assinatura (mover) : Azure for Students

ID da Assinatura : 5f05f7d7-226e-4ebb-b13b-b9605f22ab76

Marcações (editar) : Adicionar marcas

SKU : Standard

Tipo de SO : Linux

Endereço IP (Public) : 4.156.22.222

FQDN : restaurante-api.eastus.azurecontainer.io

Contagem de contêineres : 1

CPU

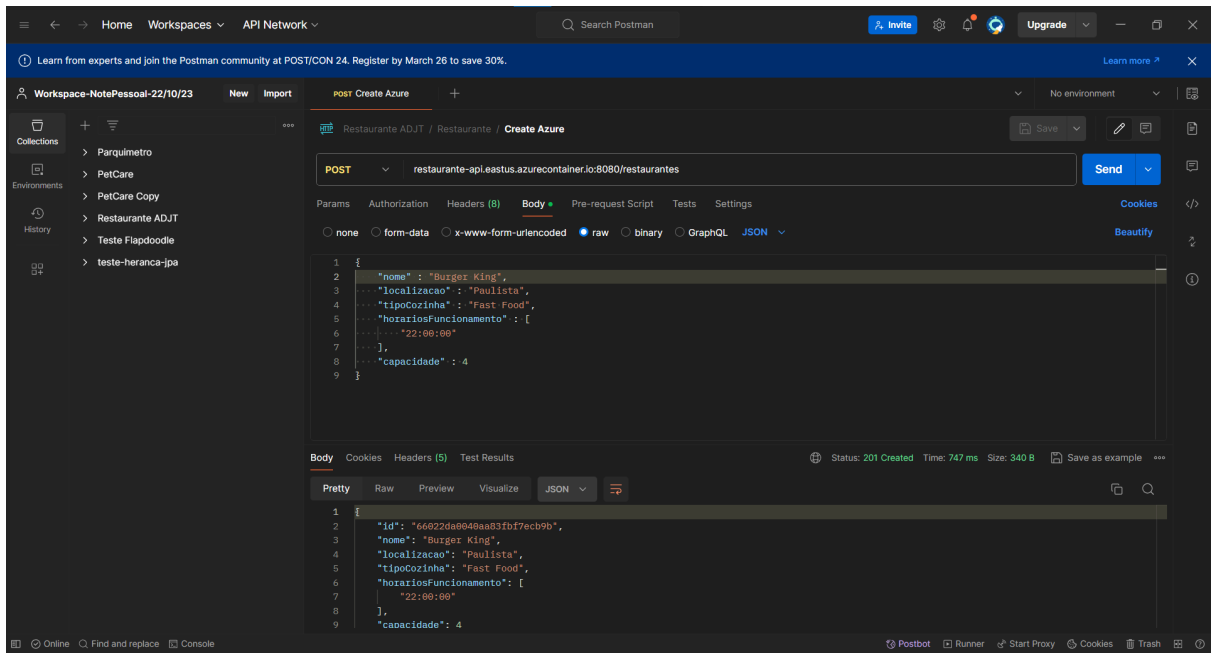
Memória

UTC-03:00

Agora os endpoints da aplicação podem ser testados através da URL:

[restaurante-api.eastus.azurecontainer.io:8080/](http://restaurante-api.eastus.azurecontainer.io:8080/)

Demonstração Criação Restaurante:



## Demonstração Listagem Restaurantes:

