



Desafio de Observabilidade da O2B Academy

Atividade Prática Observabilidade

Laboratório aplicado e orientado por **Prof.: Patrick J. Cardoso** - Treinamento de Observabilidade - O2B.

Aluno: Aguinaldo Américo

GitHub do Lab: github.com/aguinaldo1/Lab_Desafio-Observ

LinkedIn: [linkedin.com/in/aguinaldo-americo](https://www.linkedin.com/in/aguinaldo-americo)

Objetivo:

- Monitoramento com uso das ferramentas: Prometheus, Grafana e Alertmanager.
- Laboratório: Simulação de Error, Cálculo de Duração e Métricas.

Cenário: A execução deste laboratório tem por objetivo a construção de um ambiente controlado para ativação e monitoramento de cenários onde simulamos nesse caso específico e submetemos nossa aplicação a erro sistêmico e de latência a fim de buscarmos a redução de tempo médio e detecção de incidentes para que não haja impacto de interatividade e indisponibilidade em nossa aplicação.

Objetivo principal: Esse lab. tem por objetivo principal medir o conhecimento adquirido através do treinamento de observabilidade disposto pela O2B Academy. Onde sua principal proposta é subir via container uma infra contendo uma aplicação web preparada e instrumentada com métricas de erro e latência a fim de ser colhida via Prometheus, visualizada através de Dash no Grafana e recorrente a situação de erro ser alarmada através de Alertmanager com o acionamento via e-mail.

Arquitetura da Aplicação:

Aplicação Python:

- A aplicação Python está em execução em contêineres gerenciados pelo Docker Compose.
- Os contêineres incluem um serviço de servidor web - Flask.

Observabilidade com Prometheus:

- O Prometheus está coletando métricas da aplicação usando o endpoint de métricas já instrumentadas na aplicação.
- As métricas incluem o desempenho da aplicação, tempos de resposta, e erros.

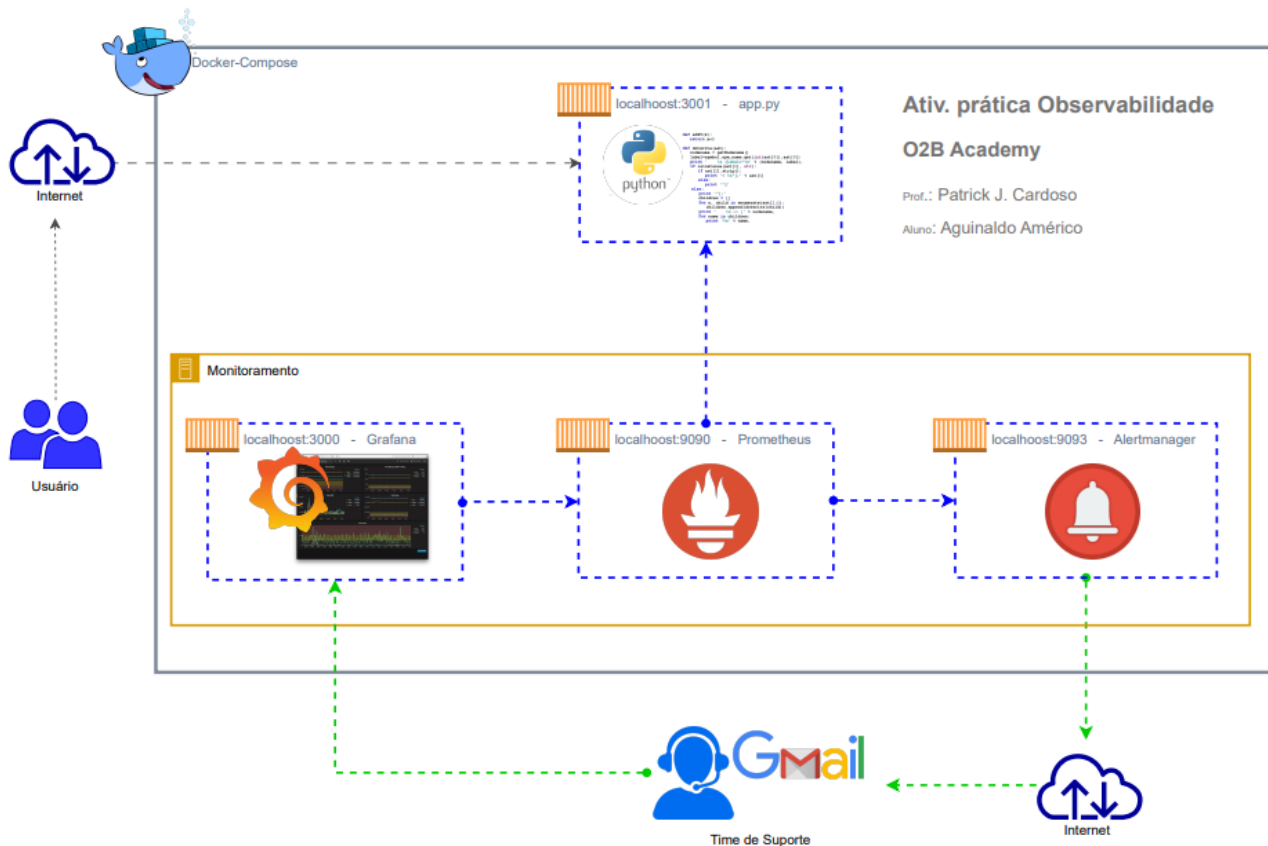
Grafana para Visualização:

- O Grafana visualiza e cria dashboards com as métricas coletadas pelo Prometheus.
- Com os Dashboards monitoramos o desempenho e a saúde da aplicação.

Alertas no Alertmanager:

- O Alertmanager foi configurado para receber alertas do Prometheus com base nas regras definidas.
- Configuramos regras para acionar alertas quando as métricas atingirem um limiar crítico que é a quantidade de erro >5.
- O Alertmanager envia notificações por e-mail configurado via gmail.

Fluxograma:



Fluxo de Observabilidade e Alertas:

Containers Docker:

O Docker Compose gerencia a orquestração dos contêineres da aplicação e do monitoramento.

Aplicação Python:

- Gera as métricas de latência e erros.

Prometheus:

- Coleta métricas da aplicação.

Grafana:

- Grafana consulta o Prometheus para exibir métricas em dashboards personalizados.

Alertmanager:

- Recebe alertas do Prometheus baseados nas regras configuradas.
- Envia notificações para a equipe quando um alerta é acionado.

Resultados:

Métricas:

- A **latência** está relacionada ao tempo de resposta na execução da aplicação.
- **erros** - quantidade de erros que estamos tendo no decorrer da execução da aplicação.

Rodando o laboratório em docker-compose temos uma stack que consiste em:

1. na camada de aplicação contém um container que possui nossa aplicação web app.py, com as devidas métricas configuradas onde geramos algumas interações com erros para que possamos ter visibilidade de forma controlada na nossa camada de observabilidade.
2. Configuramos o prometheus o grafana com seus devidos painéis (dashboards), subimos o alertmanager e configuramos a integração com o gmail como o endpoint de alertas.
3. Basicamente, o nosso Alertmanager se comunica com o gmail enviando os alertas de erro ou os eventos críticos no nosso ambiente, então ao ocorrer o incidente sendo >5 ocorrências vai gerar um alerta que será enviado via e-mail já configurado e o mesmo será visualizado por um time de suporte no qual recebe aquela mensagem e também pelos desenvolvedores dessa API.

A execução desse lab serve para simular o ambiente onde trabalharemos para a redução de tempo médio e detecção de incidentes, a fim de detectar e mitigar uma inatividade na aplicação.

Ao ter visibilidade sobre métricas e temos essa camada de observabilidade, conseguimos agir de forma proativa detectando algumas nuances dentro do nosso sistema, assim conseguimos investigar a causa raiz logo no início de uma degradação, antes que o mesmo se torne um incidente e cause uma indisponibilidade em nossa aplicação.

Evidência e passo a passo da execução do Laboratório:

1. Provisionando o Ambiente

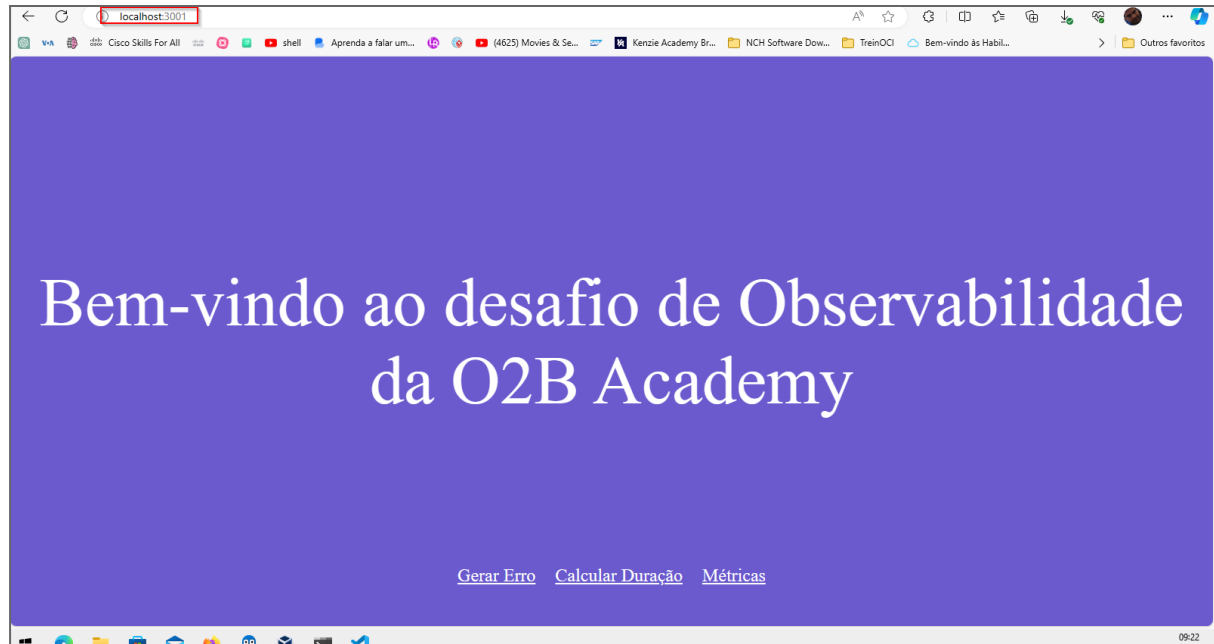
- ☒ Instalação do Prometheus e Grafana
- ☒ Criar arquivo **docker-compose.yml** para definir os serviços Prometheus e Grafana.
- ☒ Criar um diretório chamado prometheus e dentro dele arquivo **prometheus.yml** para conf. o Prometheus.

2. Configurando a aplicação de exemplo

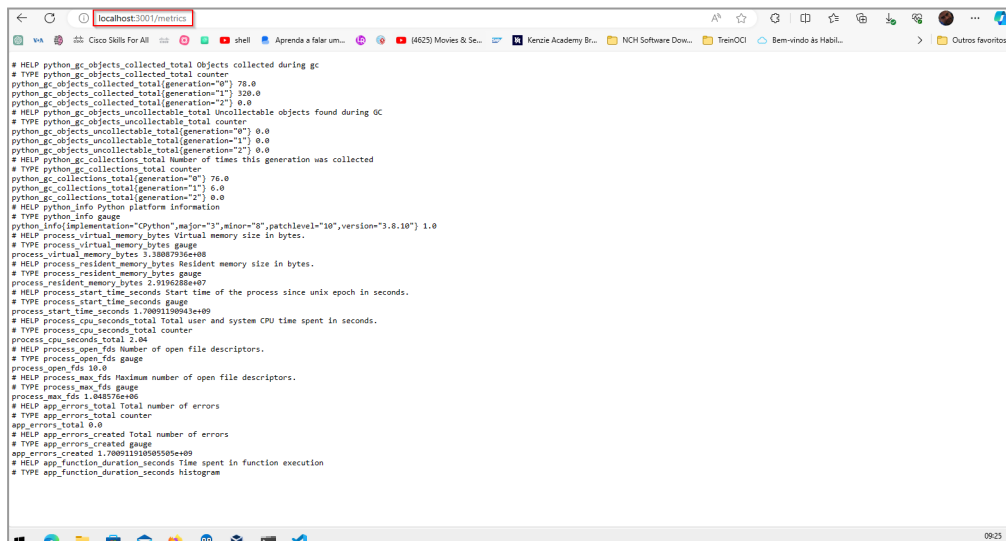
- ☒ Certifique-se de ter o Flask e Prometheus Client Python instalados: `pip install Flask prometheus_client`
- ☒ Acesse o diretório da aplicação: `cd python-app`.
- ☒ Inicie a aplicação e exponha-a em uma porta específica: `python app.py`.

3. Testando a aplicação

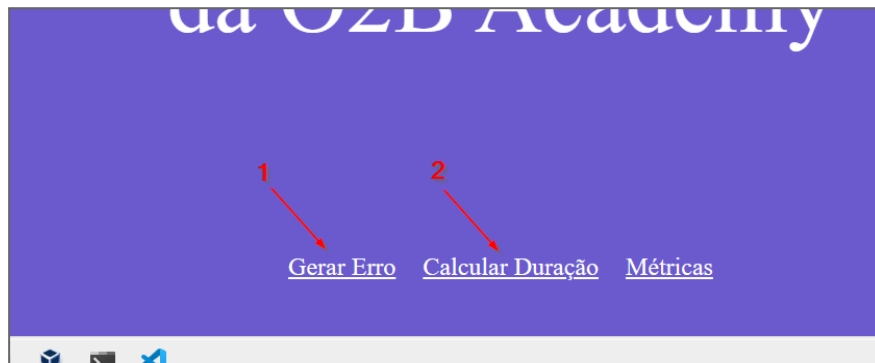
- ☒ A aplicação está disponível em: ***https://localhost:3001***



- ☒ Verificando as métricas expostas em: ***https://localhost:3001/metrics***



- ☒ Acesse a aplicação e clique em: Gerar Erro e depois em Calcular Duração.

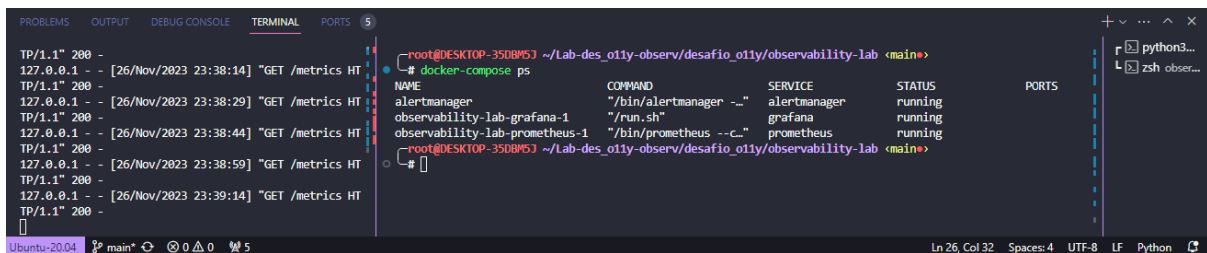


4. Configurar o Prometheus no Laboratório

- ☒ No arquivo prometheus.yml no diretório prometheus adiciona a sessão sob **static_configs** para coletar métricas da aplicação python.

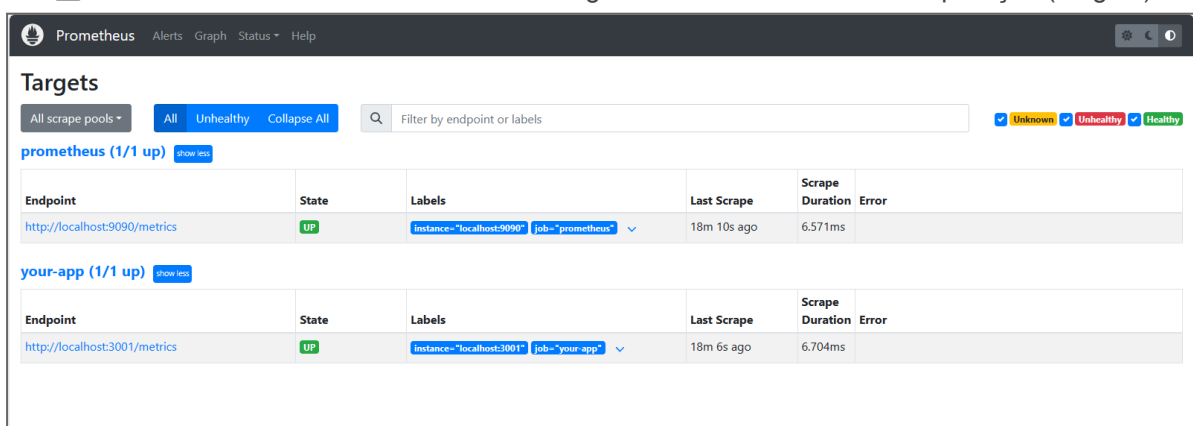
5. Iniciar o Ambiente de Observabilidade

- ☒ Iniciando os serviços Prometheus e Grafana: **docker-compose up -d**.
- ☒ Certificando-se os containers do Prometheus e Grafana subiram: **docker-compose ps**.

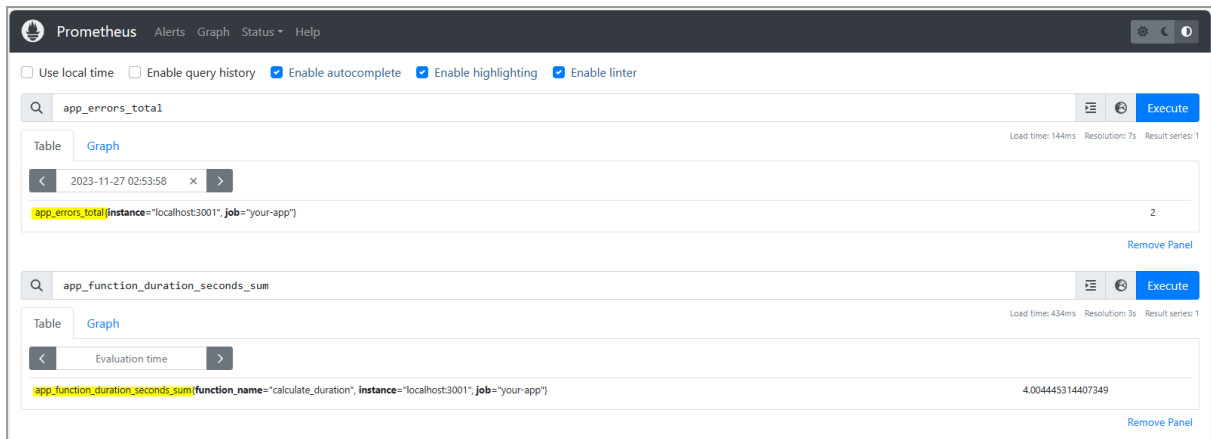


6. Acessando o Prometheus e verificando as métricas da aplicação

- ☒ Acessando o painel Prometheus no navegador em **https://localhost:9090**.
- ☒ Certificando se o Prometheus está conseguindo acessar os dados da aplicação (Targets).

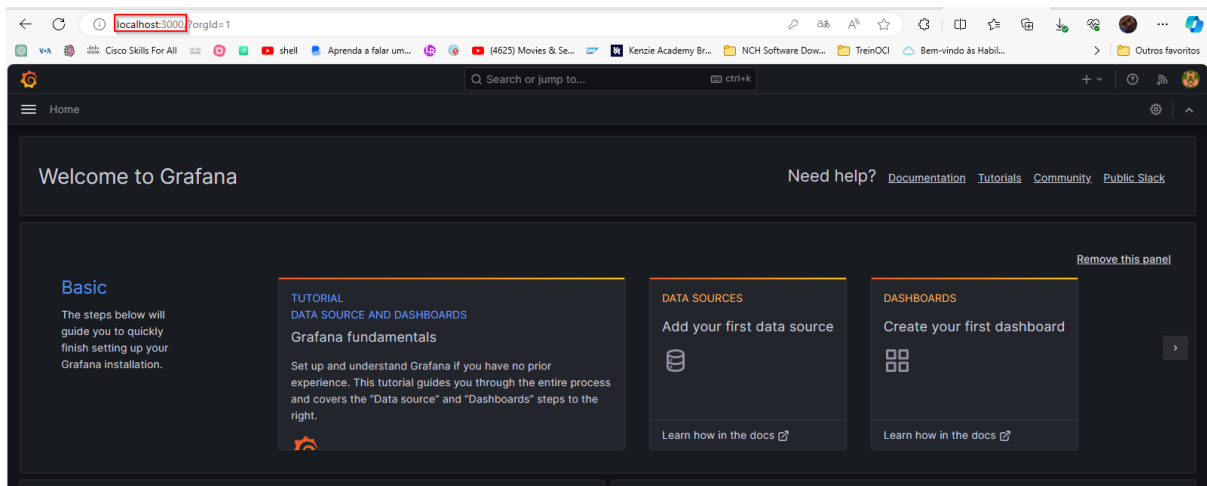


- ☒ Olhando as métricas de contagem de erros(app_errors_total) métrica que conta o número total de erros que ocorreram na aplicação.
- ☒ Olhando as métricas de duração de função (app_function_duration_seconds) métrica que mede o tempo gasto na execução de funções específicas na aplicação.



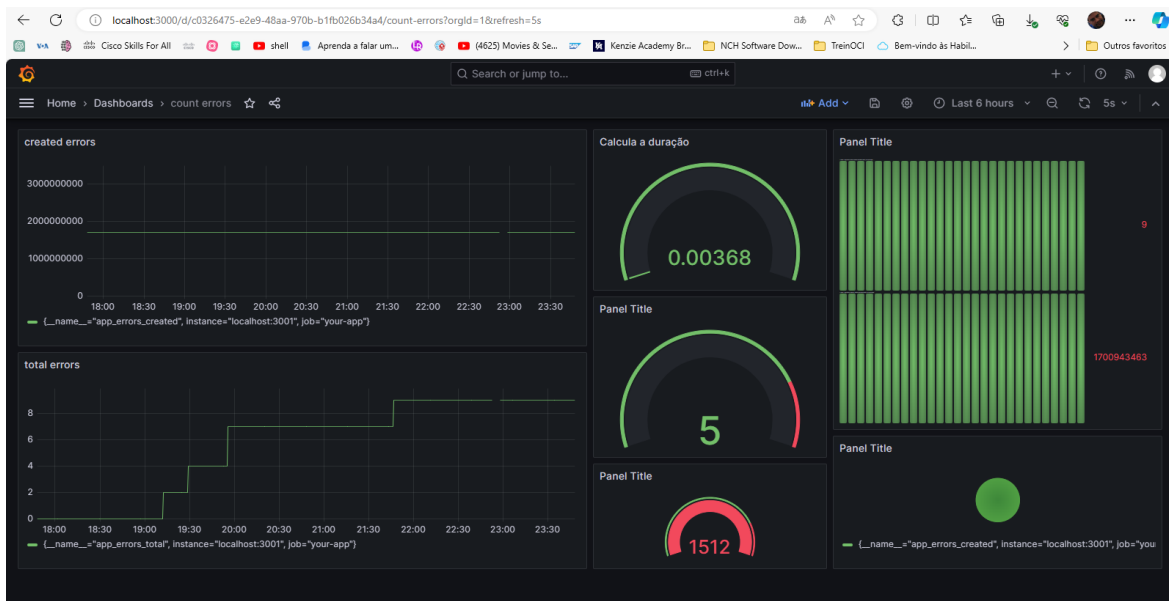
7. Configurar o Grafana

- ☒ Acessando o painel Grafana no navegador em ***https://localhost:3000***.
- ☒ Fazendo login com as credenciais padrão.
- ☒ Configurando o Prometheus como uma fonte de dados.



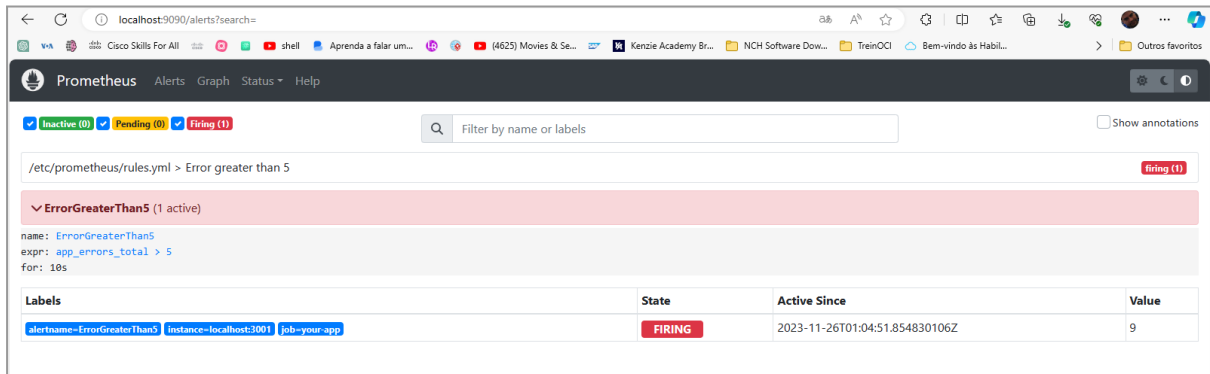
8. Criando os painéis no Grafana

- ☒ Criando o Dashboard.



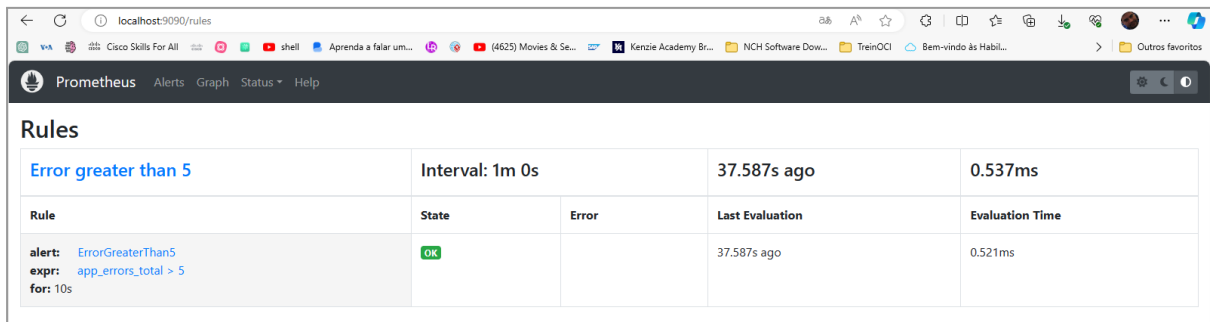
9. Configurando e gerando alerta com o Alertmanager

☑ Configurando o Alertmanager e testando no gmail.



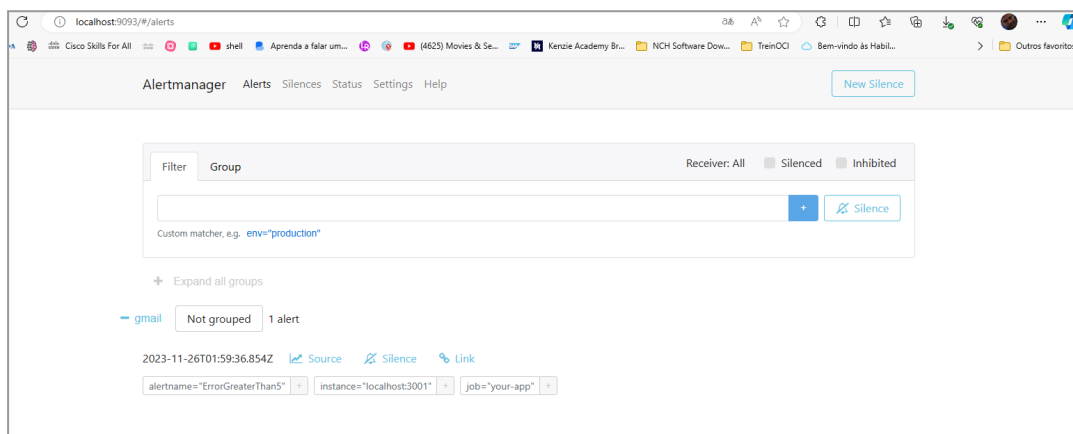
The screenshot shows the Prometheus Alerts page in a browser. The URL is localhost:9090/alerts?search=. The page displays a list of alerts, with one alert in the 'Firing' state. The alert is named 'ErrorGreaterThan5' and is active. The expression is 'app_errors_total > 5' and the for duration is '10s'. The alert is labeled with 'alertname=ErrorGreaterThan5', 'instance=localhost:3001', and 'job=your-app'. The state is 'FIRING' and the value is 9.

Labels	State	Active Since	Value
alertname=ErrorGreaterThan5 instance=localhost:3001 job=your-app	FIRING	2023-11-26T01:04:51.854830106Z	9



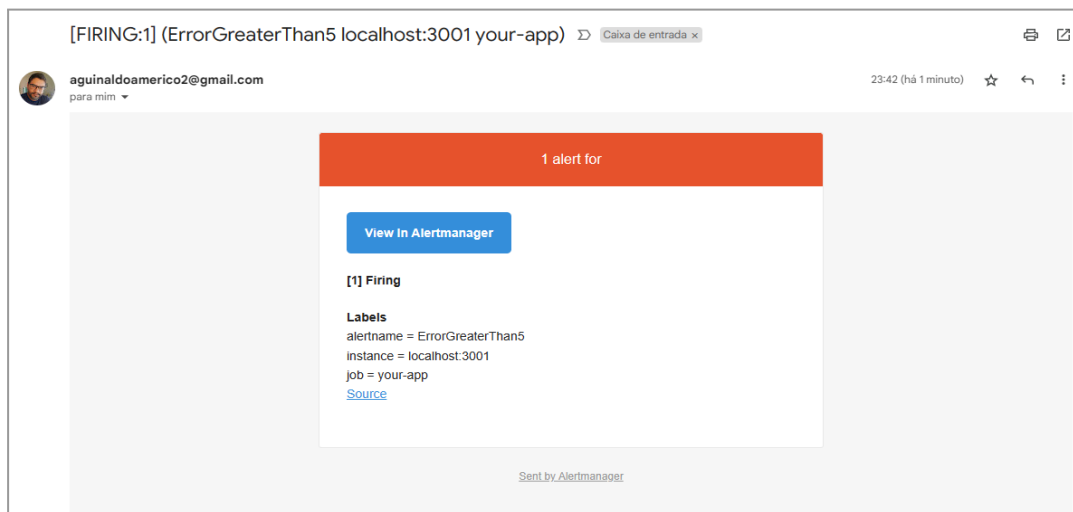
The screenshot shows the Prometheus Rules page in a browser. The URL is localhost:9090/rules. The page displays a table of rules, with one rule named 'Error greater than 5'. The rule is in the 'OK' state and has a last evaluation time of 37.587s ago. The evaluation time is 0.521ms.

Rule	State	Error	Last Evaluation	Evaluation Time
alert: ErrorGreaterThan5 expr: app_errors_total > 5 for: 10s	OK		37.587s ago	0.521ms



The screenshot shows the Alertmanager Alerts page in a browser. The URL is localhost:9093/#/alerts. The page displays a list of alerts, with one alert in the 'Firing' state. The alert is named 'ErrorGreaterThan5' and is active. The expression is 'app_errors_total > 5' and the for duration is '10s'. The alert is labeled with 'alertname=ErrorGreaterThan5', 'instance=localhost:3001', and 'job=your-app'. The state is 'FIRING' and the value is 9.

Alert	State	Labels	Value
ErrorGreaterThan5	FIRING	alertname=ErrorGreaterThan5 instance=localhost:3001 job=your-app	9



The screenshot shows a Gmail email from 'aguinaldoamerico2@gmail.com' to 'mim'. The email subject is '[FIRING:1] (ErrorGreaterThan5 localhost:3001 your-app)'. The email body contains a red banner with the text '1 alert for'. Below the banner is a button labeled 'View in Alertmanager'. The email also contains the following information:

[1] Firing

Labels

alertname = ErrorGreaterThan5
instance = localhost:3001
job = your-app
[Source](#)

Sent by Alertmanager