



# Observabilidade no Frontend com Grafana Faro.

Cícero Feijó – *Software Engineer Specialist / Sicredi*





## Sobre mim:



Engenheiro de  
Software  
Especialista

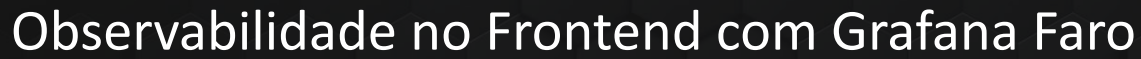
Desenvolvedor há  
mais de 20 anos.

Atuei em empresas  
como Terra, HP &  
Azion.

Instrutor há mais de  
15 anos.

Tenho um canal com  
mais de 2 mil  
inscritos.





- cerca de 1000 desenvolvedores.
- aproximadamente 10% é dev front end.
- ~15k repositórios de código.
- picos de 1.3k conexões simultâneas.
- 16 mil jobs executados por dia.

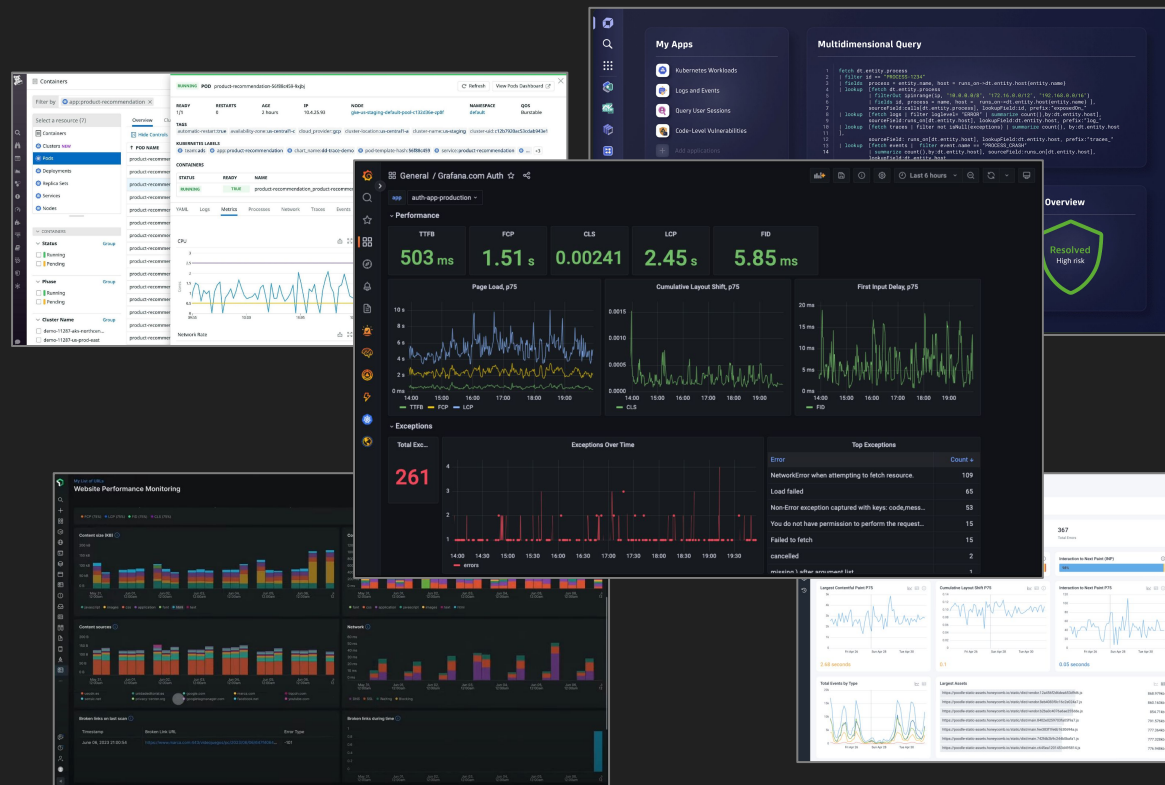


## Motivação

- Dificuldade em ter informações relacionadas a saúde da aplicação no front.
- Quando ficamos sabendo de um problema, ele normalmente já afetou algum usuário final.
- Mapeamento e reprodução de erros depende de recursos como screenshots e adivinhação.
- O erro geralmente é apresentado no console - *mas do usuário*.
- Não há uma forma de coleta de informações básicas e customizadas em um ponto central.

# Observabilidade no Frontend com Grafana Faro

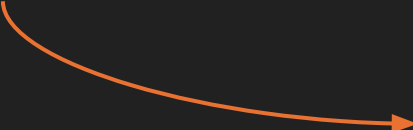
## Mercado



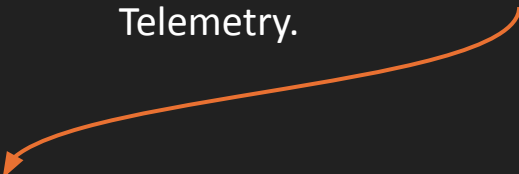


## O que é o Grafana Faro?

Uma dependência para observabilidade frontend, com uma SDK altamente configurável para monitoramento real de usuários.



Captura sinais de observabilidade do front, sendo possível correlacionar com o backend e infraestrutura via padrões de Open Telemetry.



Dessa forma proporcionando uma observabilidade “full-stack”.



## O que a dependência faz?

Realiza a coleta logs, erros, métricas de performance, traces e eventos.

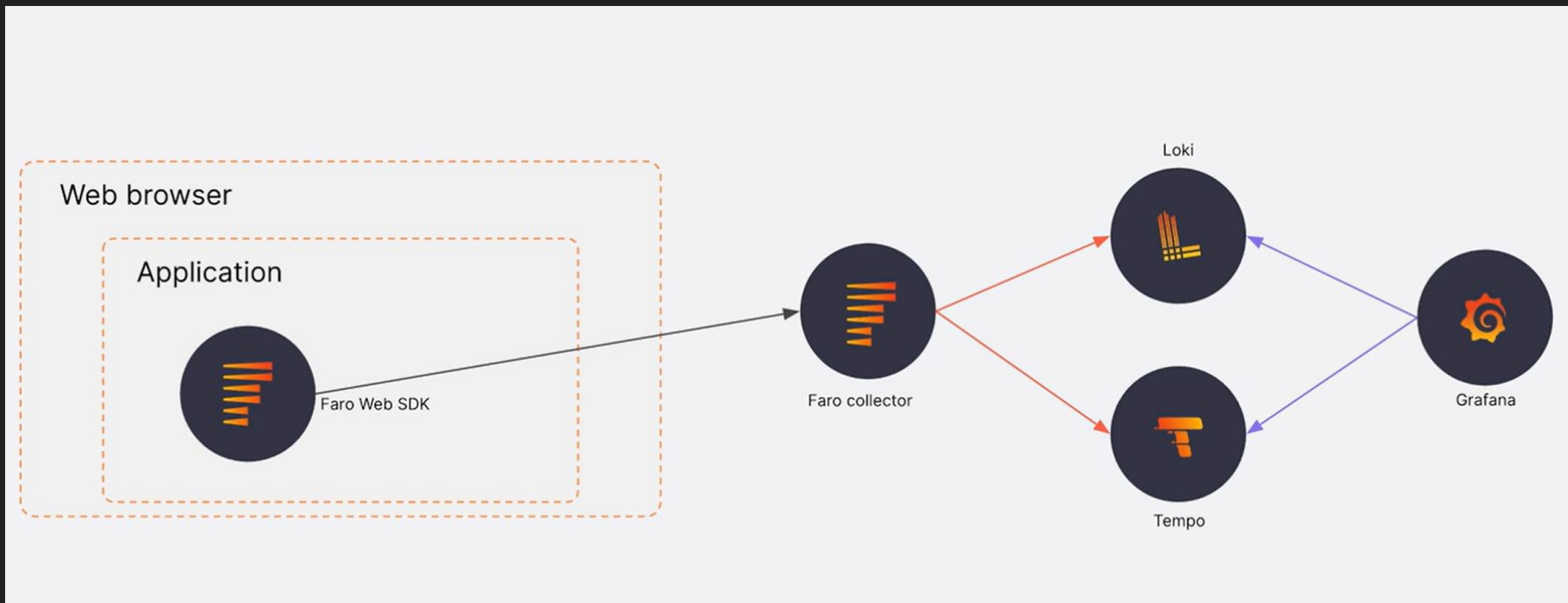
Estes dados são armazenados no Grafana Loki (*database*) e Grafana Tempo (*tracing tool*).

Como os dados podem ser relacionados com todas camadas de desenvolvimento, é possível elaborar dashboards complexas relacionadas ao trace da aplicação.





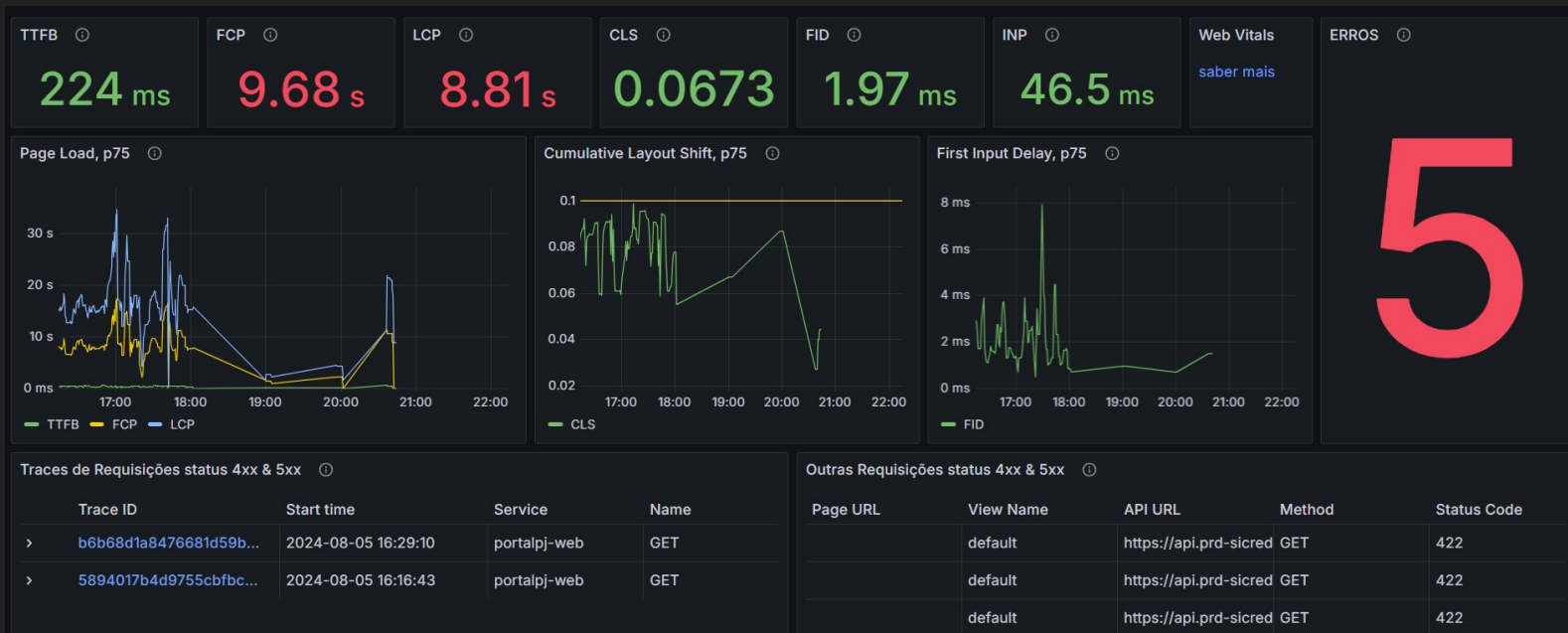
## Componentes da stack de Observabilidade





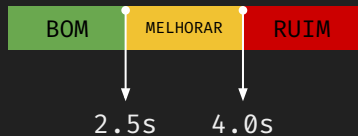
## Coleta nativa de dados

Por padrão a SDK já vem com pré-configurações de coleta para erros, logs, tracing, web vitals e session tracking de requests.

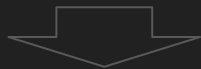




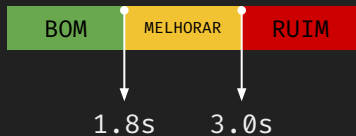
## Web Vitals - Saúde da Aplicação



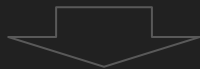
### LCP: Largest Contentful Paint



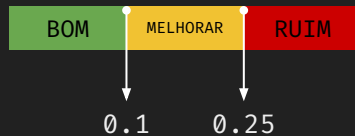
Tempo desde o início do carregamento da página até que o maior elemento de conteúdo visível seja renderizado na tela.



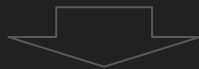
### FCP: First Contentful Paint



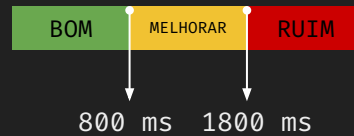
Tempo desde o início do carregamento da página até que qualquer conteúdo do DOM seja renderizado na tela.



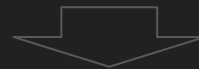
### CLS: Cumulative Layout Shift



Quantifica a soma total de todas as mudanças de layout inesperadas que ocorrem durante a vida útil da página.



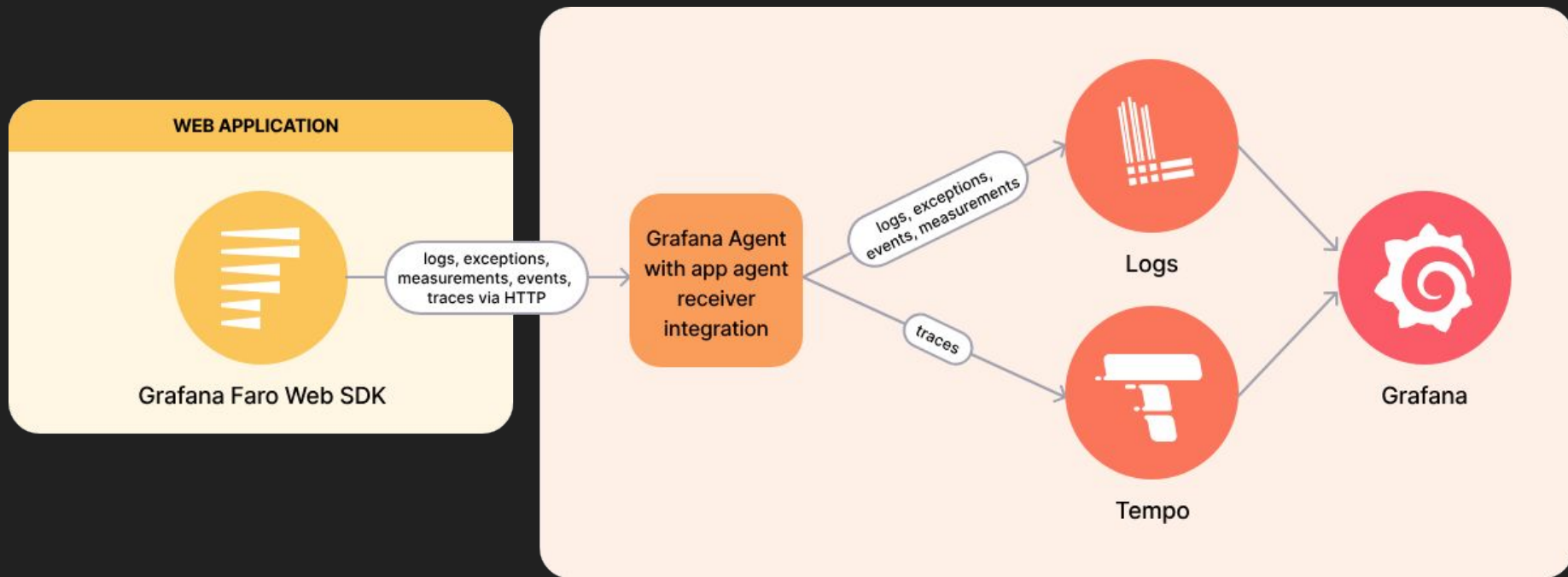
### TTFB: Time to First Paint



Tempo para o navegador renderizar pela primeira vez após a navegação, proporcionando um feedback visual inicial ao usuário.



## Fluxo do monitoramento





## Aplicabilidade

- Coleta de erros JavaScript de uma aplicação.
- Registro de logs via console.
- Mapeamento de eventos realizados pelo usuário.
- Registro de medições específicas da aplicação.
- Coleta de dados customizados por aplicação:
  - ◆ Identificação de usuário.
  - ◆ Geolocalização
  - ◆ Devices, etc.
- Tracing de API call's.



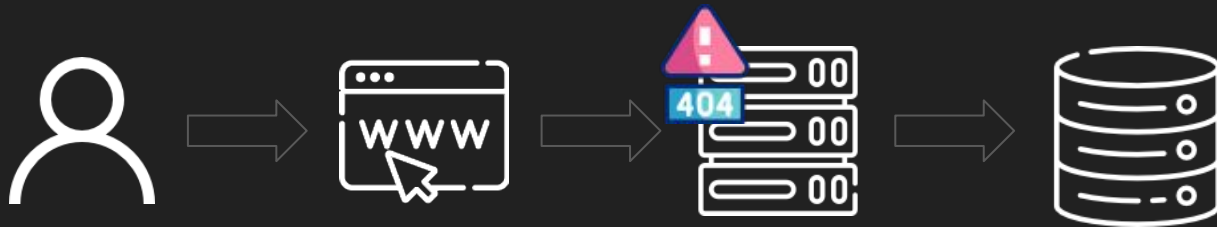


## Cenário

Entender como está o andamento da jornada de usuário em uma aplicação, no processo E2E.

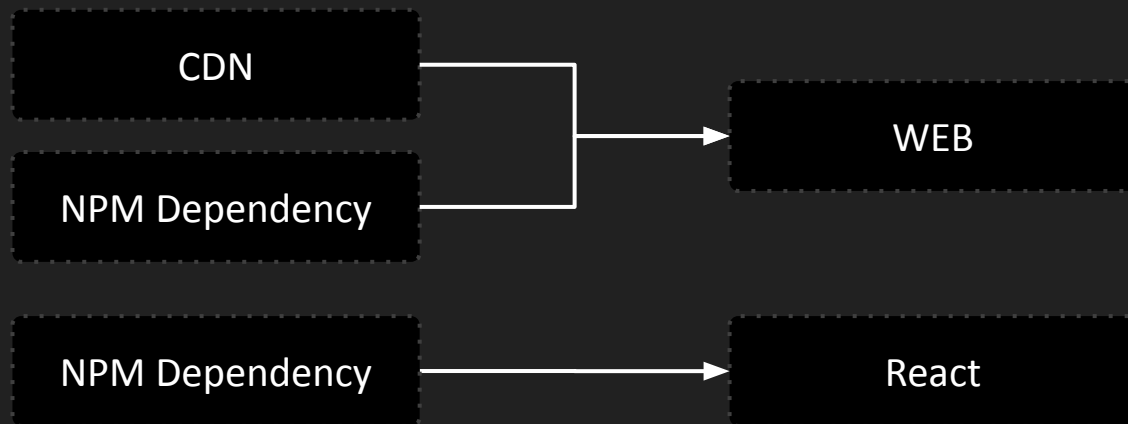
- Desde o onBoarding até a criação/configuração de algum serviço.
- criação de uma conta.
- adesão e uso.

Acompanhar performance e identificar possíveis problemas que interromperam essa jornada, como erros de JavaScript ou até mesmo retorno de API's não tratados no front.





## Implementação





## Infraestrutura

**Grafana Labs – Plano Free**  
Trial Full de 14 dias.

|                                                                    |
|--------------------------------------------------------------------|
| Community Support                                                  |
| USAGE INCLUDED                                                     |
| ✓ <b>Metrics</b><br>10k metrics billable series, 14 days retention |
| ✓ <b>Visualization</b><br>3 active users                           |
| ✓ <b>Logs, Traces, Profiles</b><br>50 GBs each, 14 days retention  |
| ✓ <b>IRM</b><br>3 active users                                     |
| ✓ <b>Application Observability</b><br>2.232k host hours            |
| ✓ <b>Frontend Observability</b><br>50k sessions                    |
| ✓ <b>k6</b><br>500 virtual user hours, 14 days retention           |
| ✓ <b>Synthetics</b><br>100k test executions                        |

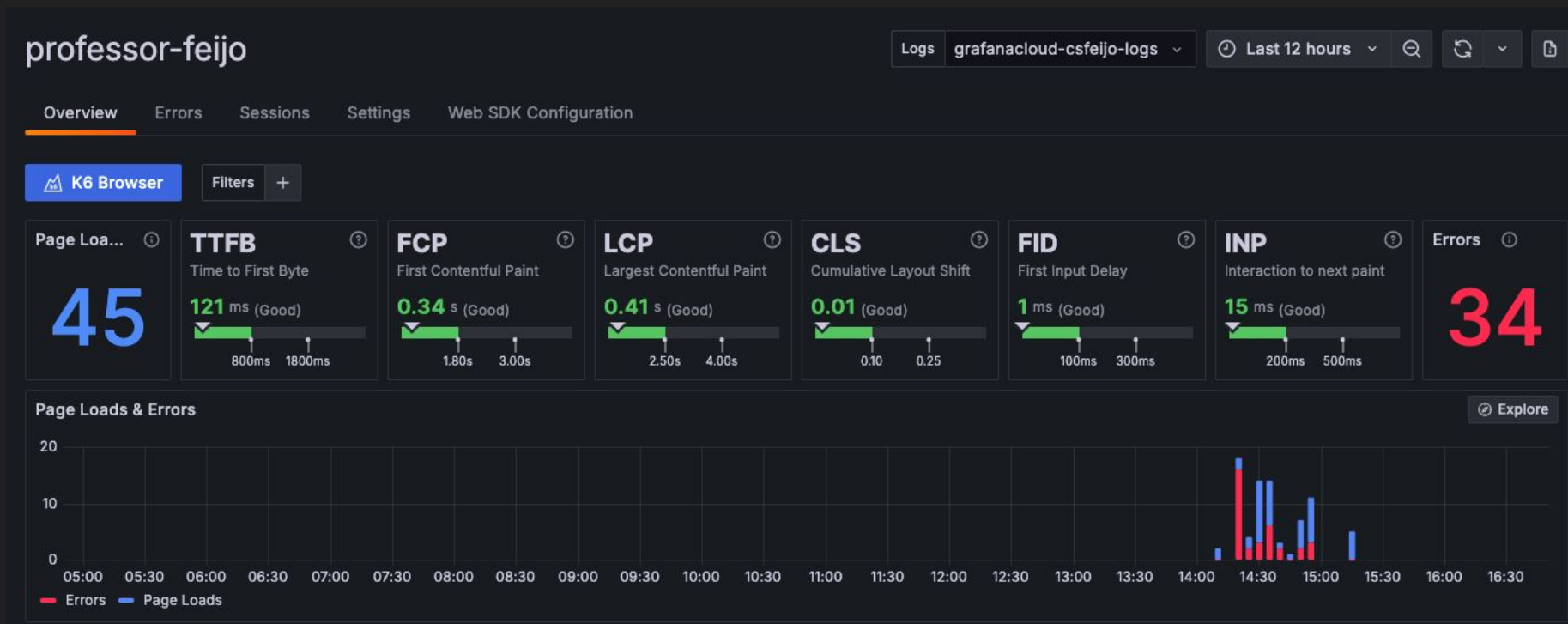
**Internal hosting**  
Infraestrutura própria.

Configuração do Collector (Alloy)  
Grafana Loki  
Grafana Tempo  
Dashboard  
Retenção conforme necessidade

Custos com infraestrutura local +  
armazenamento, backup, etc.

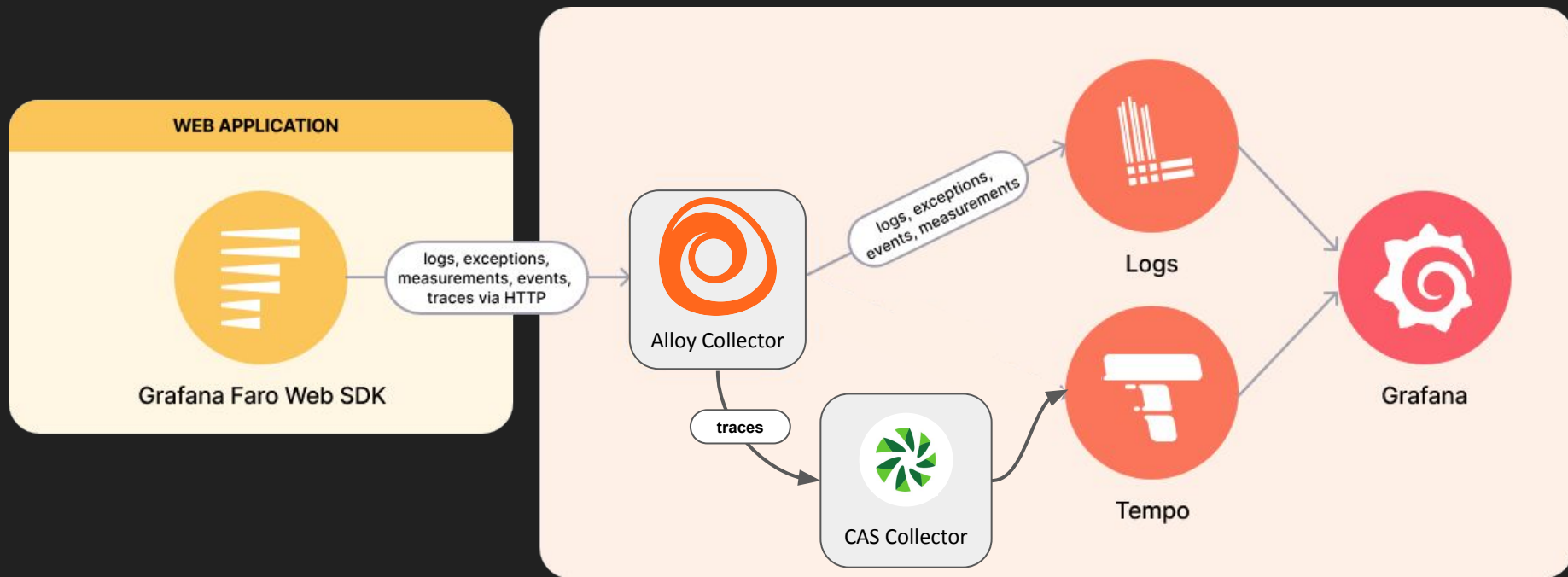


## Dashboard do Grafana Cloud





## Customizações que fizemos





## Grafana Alloy como Agent

- Novo collector de logs Open Telemetry da Grafana.
- Possibilita o tratamento de algumas informações antes de enviá-las para as outras layers.
- Sem o Alloy as aplicações FE enviariam os traces de suas informações direto para o Tempo.
- Permite a serialização dos dados vinculando-os a um *session\_id*.
- Esses mesmos *session\_id*'s são enviados nas requests para o backend.
- E isso permite a mágica de ter um trace E2E.



## Implantação

- Criamos uma lib que encapsula o Faro, mas disponibilizada via CDN.
- Produtos que rodam abaixo do NGINX incorporam a funcionalidade via “sub\_module”.
- No primeiro acesso da app (mesmo local) a dashboard já é criada.
  - ◆ É possível monitorar saúde e erros desde o desenvolvimento.



## Exemplo de payload

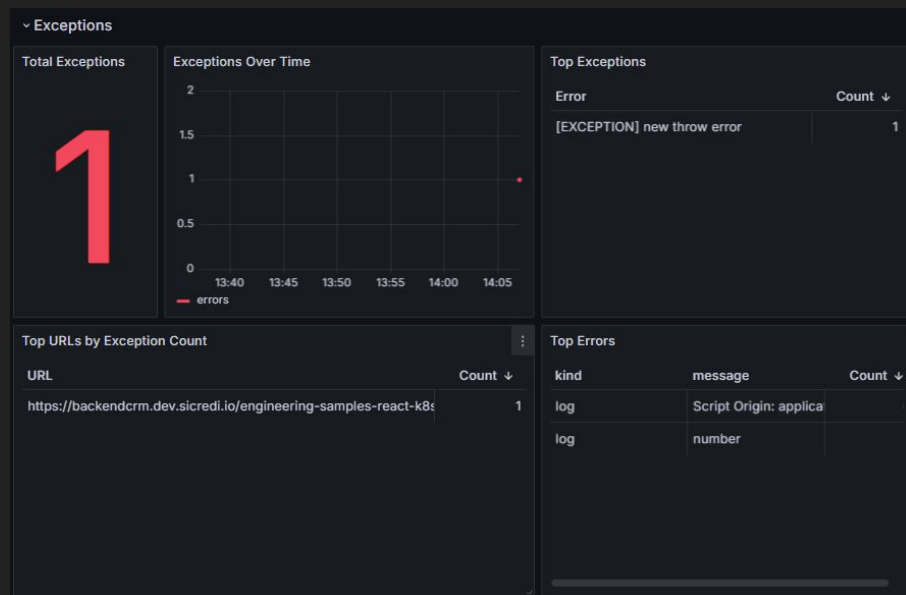
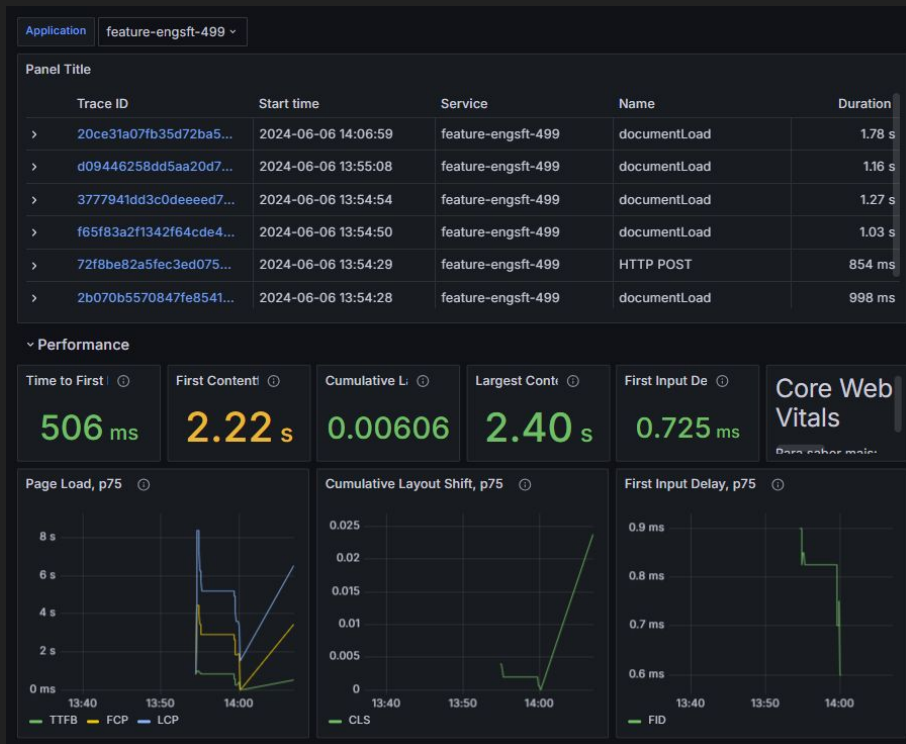
```
▼ Carga de Solicitação  Exibir código-fonte
▼ {meta: {sdk: {version: "1.9.0"},...}, events: [...]}
  ▼ events: [...]
    ▼ 0: {name: "view_changed", domain: "browser", attributes: {fromView: "default", toView: "another-route"},
      ▶ attributes: {fromView: "default", toView: "another-route"}
        domain: "browser"
        name: "view_changed"
        timestamp: "2024-08-05T00:58:07.935Z"
    ▼ meta: {sdk: {version: "1.9.0"},...}
      ▼ app: {name: "grafana-faro-demo-app", version: "1.0.0", environment: "production"}
        environment: "production"
        name: "grafana-faro-demo-app"
        version: "1.0.0"
      ▼ browser: {name: "Edge", version: "127.0.0.0", os: "Windows 10",...}
        ▶ brands: [{brand: "NotA;Brand", version: "99"}, {brand: "Microsoft Edge", version: "127"},...]
        language: "pt-BR"
        mobile: false
        name: "Edge"
        os: "Windows 10"
        userAgent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
        version: "127.0.0.0"
        viewportHeight: "922"
        viewportWidth: "964"
      ▼ page: {url: "http://localhost:5173/"}
        url: "http://localhost:5173/"
        ▶ sdk: {version: "1.9.0"}
        ▶ session: {id: "2rPAokgNMVG", attributes: {previousSession: "hq76A2h1VE"}}
        ▶ user: {email: "bob@example.com", id: "123abc", username: "bob", attributes: {role: "manager"}}
          ▶ attributes: {role: "manager"}
            email: "bob@example.com"
            id: "123abc"
            username: "bob"
      ▼ view: {name: "another-route"}
        name: "another-route"
```

Informações como evento disparado, rota modificada, entre outros podem ser registradas.

Algumas como os dados do browser são nativas da SDK.



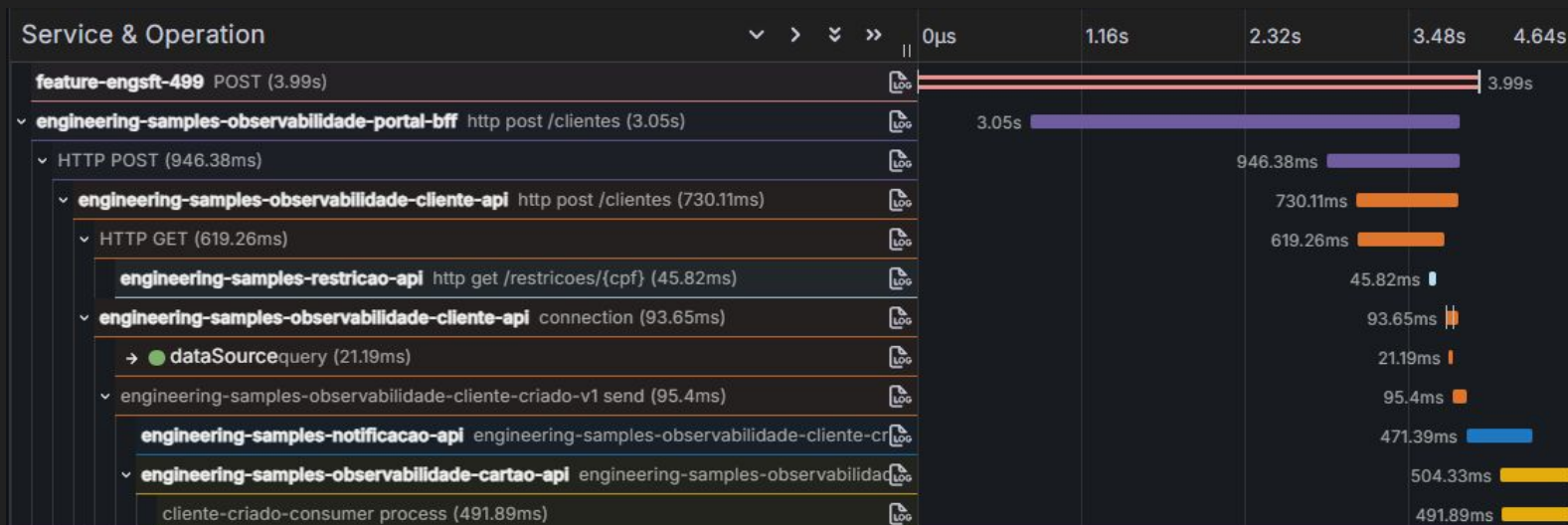
## Alguns cards que exploramos na dashboard





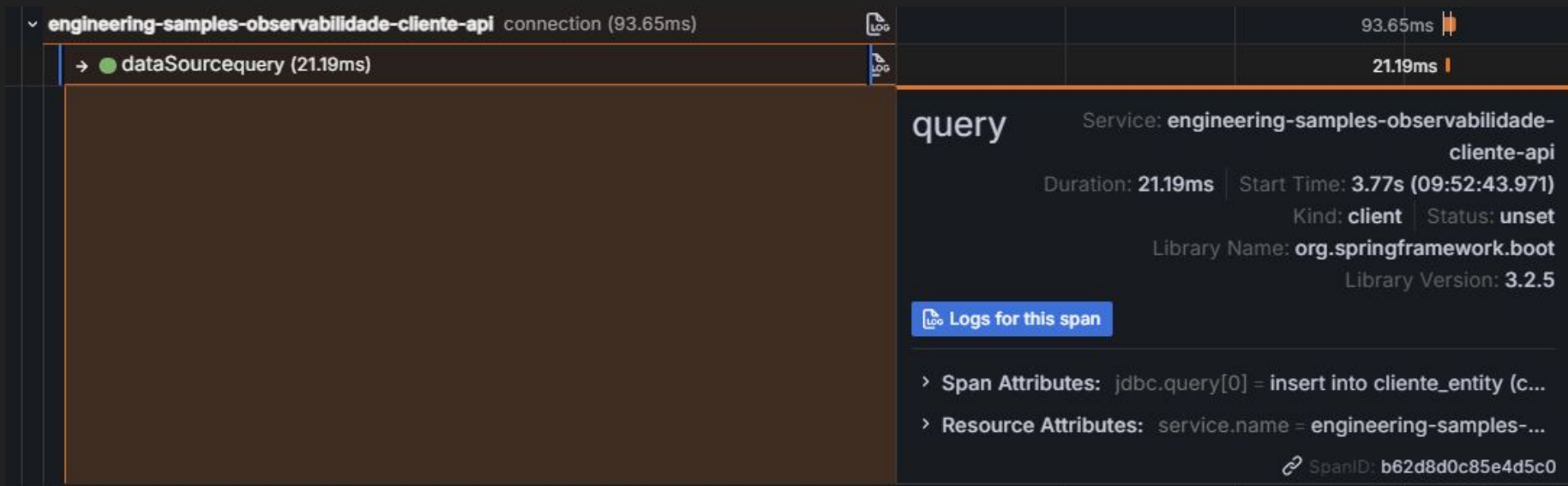
## Tracing de API's

| Panel Title |                        |                         |                    |           |          |
|-------------|------------------------|-------------------------|--------------------|-----------|----------|
|             | Trace ID               | Start time              | Service            | Name      | Duration |
| >           | 4e45c4180a05cce5a4...  | 2024-06-07 09:52:40.200 | feature-engsft-499 | POST      | 4.64 s   |
| >           | f2eeb25766e9f087ae9... | 2024-06-07 09:52:07.151 | feature-engsft-499 | HTTP POST | 526 ms   |





## Tracing de API's



The screenshot displays the Grafana Faro tracing interface. On the left, a tree view shows a span named 'engineering-samples-observabilidade-cliente-api' with a duration of 93.65ms. It is expanded to show a sub-span 'dataSourcequery' with a duration of 21.19ms. The main area on the right shows the details for the 'dataSourcequery' span, which is a 'query' operation. The details include the service name, duration, start time, kind, status, library name, and library version. A button 'Logs for this span' is visible. Below the button, the span attributes are listed, showing the SQL query and the service name. The span ID is also displayed at the bottom right.

| Span Name                                                  | Duration |
|------------------------------------------------------------|----------|
| engineering-samples-observabilidade-cliente-api connection | 93.65ms  |
| → dataSourcequery                                          | 21.19ms  |

**query**

Service: **engineering-samples-observabilidade-cliente-api**

Duration: **21.19ms** | Start Time: **3.77s (09:52:43.971)**

Kind: **client** | Status: **unset**

Library Name: **org.springframework.boot**

Library Version: **3.2.5**

[Logs for this span](#)

Span Attributes: jdbc.query[0] = insert into cliente\_entity (c...

Resource Attributes: service.name = engineering-samples-...

SpanID: b62d8d0c85e4d5c0



## Material de auxílio



### **Dockerfile para setup de todo ambiente**

<https://github.com/Condla/web-shop-o11y-demo/blob/main/docker-compose.yml>



### **[ Vídeo ] Mastering Frontend Observability in React with Grafana Faro**

[https://www.youtube.com/watch?v=IA\\_-zkpVhIU](https://www.youtube.com/watch?v=IA_-zkpVhIU)



### **Repositório com um sample para uso:**

<https://github.com/csfeijo/grafana-faro-demo-app>

# Perguntas?

Contato:  
[professorfeijo.com.br](https://professorfeijo.com.br)

